
# Argentum Online
# **Manual de Proyecto**

Taller de Programación I
Primer Cuatrimestre 2020

**Mauro Parafati** - 102749
**Santiago Klein** - 102192
**Yuhong Huang** - 102146

# Índice

# 1. Integrantes

Se listan a continuación los integrantes del proyecto:


- **HUANG, Yuhong** - Padrón: 102146 - Mail: yhuang@fi.uba.ar.
- **KLEIN, Santiago** - Padrón: 102192 - Mail: sklein@fi.uba.ar.
- **PARAFATI, Mauro** - Padrón: 102749 - Mail: mparafati@fi.uba.ar.


# 2. Enunciado

El trabajo final consistió en la elaboración de un *remake* del conocido juego **Argentum Online** desarrollado por Pablo Ignacio Márquez y publicado en 1999, exclusivamente para sistemas operativos Microsoft Windows. El mismo es un juego de género MMORPG *(massively multiplayer online role-playing game)*, en el que cada jugador controlará diferentes personajes con distintas razas y clases en un mundo lleno de magia y fantasía.

Para su implementación, se exige la utilización de **sockets TCP bloqueantes**, del paradigma de **Programación Orientada a Objetos**, así como de la **programación concurrente** mediante el uso de **múltiples hilos de ejecución**.

Con el objetivo de no extender de manera innecesaria este manual, se provee acceso indirecto al enunciado: el mismo se puede encontrar en la sección **Documents** del repositorio del proyecto:

```
https://github.com/mauro7x/taller_final
```

# 3. División de tareas

La división de tareas fue realizada teniendo en cuenta la experiencia previa de cada integrante en cada uno de los tópicos, así como el interés de cada uno por realizar partes específicas del proyecto. A grandes rasgos, se dividó al equipo en dos de la siguiente forma:

- **Santiago Klein** y **Yuhong Huang** se dedicaron al diseño y el armado del **servidor**, desde el modelado del juego *(actores principales, polimorfismos utilizados, patrones de diseño útiles)*, hasta la programación de los distintos componentes de control del mismo *(mecanismos de conexión, bases de datos, sistemas de comunicación con los clientes)*.
- **Mauro Parafati** por otro lado se encargó del **cliente**, desde la comunicación con el servidor hasta el diseño de la interfaz gráfica *(máquina de estados, widgets visuales, formularios, interacción con el usuario, animaciones de movimiento, sonidos)*.

Esta división no implica de ninguna manera que ambos modelos no hayan sido diseñados por todo el equipo, sino todo lo contrario. Se propuso desde un primer día, reuniones periódicas en las que cada integrante expondría sus ideas y sus progresos, a fin de obtener *feedback*, críticas y posibles mejoras al modelo por parte de los demás, fomentando la colaboración mutua y la permanente comunicación para la evolución integrada de las distintas features del juego.

# 4. Evolución del proyecto

La primer semana se dedicó a la puesta en común de ideas de los integrantes sobre el problema a resolver, con el objetivo de definir el diseño y el esquema general de ambos aplicativos (**cliente y servidor**). Una vez finalizado este proceso, se comenzó con la división previamente detallada a diseñar los mecanismos complejos de cada parte, intentando establecer objetivos semanales.

## 4.1. Cronograma del servidor

Para el desarrollo del **servidor**, se siguió el siguiente cronograma:

- **Semana 1:** boceto del diseño general de las clases que intervienen en el juego, demarcación y división de responsabilidades de las mismas, identificación de hilos necesarios para la conexión con muchos cientes. Puesta en común de features necesarios para la comunicación con el cliente.
- **Semana 2:** implementación del proxy de comunicación con múltiples clientes. Elaboración del motor principal, y las clases auxiliares para la conexión de un nuevo cliente. Puesta a prueba con cliente proxy. Paralelamente, desarrollo de la lógica del juego.
- **Semana 3:** definición del protocolo e implementación de los broadcasts.
- **Semana 4:** implementación de la persistencia, colaborar log-in con la persistencia implementada. Paralelamente, desarrollo de la lógica del juego e implementación progresiva de los comandos.
- **Semana 5: primer entrega.** Lógica de las criaturas, testeo y corrección de errores. Agregado de eventos.
- **Semana 6:** Documentación, refactorización del código. Agregado de nuevas features personalizadas en conjunto con el cliente.
- **Semana 7: entrega final.** Documentación final, correcciones, arreglo de errores.

## 4.2. Cronograma del cliente

En cuanto al desarrollo del **cliente**, el cronograma propuesto fue el siguiente:

- **Semana 1:** draft del cliente, armado de la primer ventana, renderizado de un mapa basado en tiles y de un personaje que se mueva por el mismo.
- **Semana 2:** renderizado de los personajes 'por partes', movimiento del mismo con un servidor-proxy, renderizado del mapa por 'filas' para lograr la correcta superposición de las imágenes.
- **Semana 3:** diseño de la hud y de la consola, input parser para la consola de comandos.
- **Semana 4:** implementación de los handlers para los eventos de i/o del usuario, como los clicks, el texto, etc. Implementación de distintos comandos que se envían al servidor.
- **Semana 5: primer entrega.** Sonidos, máquina de estados para la creación de ventanas de log-in y de crear personaje, testeo y arreglo de errores.
- **Semana 6:** Documentación, diseño de más mapas, de más monstruos. Adición de efectos especiales para los hechizos y otros, mensajes flotantes para el chat general.
- **Semana 7: entrega final.** Documentación final, correcciones, arreglo de errores, armado del video trailer del juego.

Gracias a la dedicación, responsabilidad y compromiso del equipo para con el proyecto, se pudo cumplir el cronograma propuesto. A su vez, creemos que ayudó proponer un cronograma que no sea ambicioso en exceso, para evitar sentir presión desde un principio. Por el contrario, se intentó separar cada etapa en sub-etapas más pequeñas para así lograr pequeños avances cada día.

# 5. Inconvenientes encontrados

## 5.1. Inconvenientes grupales

El principal inconveniente con el que se encontró el grupo tuvo lugar la primer semana: el día de presentación del proyecto, se nos informó que se agregaría un cuarto integrante debido a que el mismo no tenía grupo hasta el momento, cambiando los planes previos de división de tareas que teníamos hasta el momento. Tras una primer semana de re-estructuraciones y trabajo conjunto de integración, este cuarto miembro quedó fuera de la materia, por lo que tuvimos que realizar cambios importantes en la división de tareas y re-asignaciones de las mismas, que fueron costosas en tiempo.

Resultó desafiante el hecho de tener que diseñar la comunicación cliente-servidor de manera que se ajuste a nuestras necesidades eficientemente. No obstante, consideramos que supimos plantear un modelo sólido y extensible de manera muy fácil, cosa que pudimos comprobar en las últimas semanas al agregar nuevas features sin cambios significativos.

## 5.2. Inconvenientes personales

Se detallan a continuación los problemas de índole personal que cada miembro tuvo a lo largo del desarrollo del proyecto.

- **Mauro Parafati:** el inconveniente más importante que tuve no tuvo que ver con el problema en sí, sino con que que se me hizo muy difícil manejar la presión del trabajo en general, así como sus tiempos. Por momentos me sucedía que si no lograba avances en dos días, sentía que no íbamos a llegar y me desesperaba. Gracias a la responsabilidad de mis compañeros y al trabajo diario que los tres le dedicamos, pude superar estas presiones y me siento muy satisfecho con el trabajo final. Por otro lado, en cuanto a inconvenientes técnicos, al estar tratando con temas completamente nuevos para mi (renderizado de sprites, el armado de un juego como tal), muchas veces me sucedió que no sabía como empezar a encarar un problema, ni que enfoque darle al mismo, pero lo pude resolver muchas veces buscando ejemplos y muchas otras probando en *ejemplos de juguete* lo que quería desarrollar.
- **Santiago Klein:** el hecho de lidiar por primera vez con un proyecto de tamaño considerable provocó, en un principio, cierta incertidumbre con la manera de manejar los tiempos y el diseño general del proyecto. No obstante, un factor esencial fue la constante comunicación con el comprometido grupo, que allanó el camino frente a incovenientes y dudas en el manejo de los temas nuevos vistos en la materia, así como del lenguaje *C++*.
- **Yuhong Huang:** el primer inconveniente que tuve sucedió al principio del trabajo, puesto que se me hizo difícil arrancar con un proyecto tan grande, con tantos temas. Me sentí muy perdido en la primer semana. Otro inconveniente que tuve a lo largo del desarrollo, tuvo que ver con la ansiedad que tengo, razón por la cual muchas veces cometí varios errores básicos por programar rápido. También por momentos resolvía los problemas con soluciones más complicadas de lo necesario, lo cual generaba más errores, y perdida de mucho tiempo para encontrarlos. Con la ayuda de mis compañeros y sus paciencias, pude encontrar los errores que cometí y a simplificar soluciones complejas.

### 5.2.1. Inconvenientes técnicos

Se llegó a la primer entrega con un trabajo prácticamente terminado: todas sus features se encontraban funcionales. Sin embargo, nos encontramos con el primer inconveniente técnico del proyecto: **un evelado consumo de CPU**, que sólo se podía apreciar en computadoras con bajos recursos que realizaran el renderizado por software. Frente a esto, se procedió a intentar encontrar el origen del mismo utilizando **KCacheGrind** de Valgrind, que proporciona un profiler muy útil

que básicamente nos dice en qué funciones nuestro programa estuvo un mayor tiempo ejecutándose, y qué tanto de este tiempo se estuvo ejecutando código de este método o de métodos llamados por el mismo.

Gracias a esta herramienta, descubrimos que teníamos un gran problema en el loop de renderizado: como nuestro mapa estaba almacenado en una matriz, para renderizar sus capas era necesario recorrer la misma. Se estaba recorriendo toda la matriz (algoritmo $O(n^2)$) para luego chequear si cada tile era visible por la cámara. Si esta última verificación era positiva, se procedía al renderizado. Si bien no se estaban renderizado imágenes de más (lo que hubiese sido mucho más notorio desde un principio), se estaban realizando muchos chequeos innecesarios, que crecían cuadráticamente con el tamaño del mapa.

La **solución** encontrada fue acotar el ciclo *for* que recorre la matriz del mapa al rango visible por la cámara, preguntándole de antemano cuáles eran estos límites a la misma. De esta forma, si nuestro mapa fuera de $1000x1000$, se pasaría de realizar 1000000 chequeos a realizar un número constante de chequeos, definidos por la cantidad de tiles que la cámara le muestra al jugador.

Si bien esto ayudo mucho, no solucionó completamente el problema pues el consumo seguía siendo elevado en máquinas con pocos recursos. Para proveer de una solución un poco menos bonita pero eficiente, se diseño un sistema de configuración de los **cuadros por segundo** *(fps)* que se le muestran al usuario, para que si su máquina contaba con pocos recursos estos se pudieran bajar de los 60 originales, a por ejemplo, 30. Con esta técnica se logró reducir el consumo entre un 20 y un 40 % aproximádamente, dependiendo de cada máquina.

# 6. Análisis de puntos pendientes

Si bien el proyecto que se plantea en el enunciado fue **resuelto en su totalidad**, a lo largo del desarrollo del mismo surgieron distintas ideas/optimizaciones para ser implementadas en un futuro. Se listan a continuación algunas de ellas:

- **Scrolling en la consola.** Mejora que consideramos de muy baja prioridad, puesto que no imposibilita ningún accionar en el juego, simplemente agrega comodidad.
- **Vista previa de información de los objetos.** Mejora de jugabilidad. La idea consiste en que cuando se pase el mouse por encima de un objeto, se nos muestre una pequeña ventana con información sobre el mismo.
- **Dividir algunos archivos de configuración.** Por ejemplo, el archivo de configuración del cliente (config.json) es demasiado grande y tiene mucha información. Se podría separar el mismo en distintos sub-archivos para cada parte del cliente.
- **Refactorizar el sistema de movimiento en el cliente.** Se utilizó un sistema basado en tiempo real para determinar cuántos pixeles debe avanzar una unidad. Si bien funciona correctamente, en caso de que la función para obtener el tiempo tenga algún tipo de delay, esto podría notarse. Para optimizar el mismo, se propone utilizar las iteraciones del cliente como medida de tiempo transcurrido para avanzar la posición de la unidad en el mapa.
- **Indicador que diga la posición actual en una esquina.** Mejora de muy baja prioridad para ganar jugabilidad.
- **Globo de mensajes generales.** Según el mapa, se pueden encontrar fondos que dificulten la lectura de los mensajes. Se propone para solucionar este problema, el uso de globos que contengan los mensajes y contraste con los mismos para facilitar la misma en cualquier contexto.

- **Mejoras en el balance.** Si bien se intentó balancear el juego lo mejor posible, la mayoría de valores se manejaron como absolutos, como es el caso por ejemplo de las pociones de vida (curan 100 puntos de vida). Una mejora a realizar sería convertir estos valores a relativos según la vida máxima, así como agregar escalados de daño según el nivel a los personajes, y demás.

# 7. Herramientas

Se listan a continuación las distintas herramientas que se utilizaron en el desarrollo del proyecto:

- Para alojamiento del proyecto y control de versiones, utilizamos **Git** (en particular, **GitHub**).
- Para edición de código fuente, utilizamos **Visual Studio Code**, principalmente por su conocida feature *IntelliSense* muy útil para el desarrollo en C++, así como por su integración con sistemas de control de versiones como GitHub.
- Para generación de archivos de compilación e instalación, se utilizó la herramienta **CMake**.
- Para el depurado de los aplicativos se utilizó **GDB**.
- Para encontrar leaks de memoria, se utilizó la herramienta **MemCheck** de **Valgrind**.
- Para encontrar qué funciones estaban realizando un consumo del CPU mayor al esperado, y así solucionar problemas de performance, se utilizó el profiler **KCacheGrind** de **Valgrind**.
- Tanto para el armado de los distintos archivos de configuración, así como para la serialización de estructuras, se decidió utilizar el formato **JSON** en conjunto con la siguiente librería: **JSON for C++**, que ofrece muchas facilidades para su uso en C++. Esta librería, a su vez, incluye serialización y deserialización con **MsgPack**, utilizada para la comunicación entre cliente y servidor.
- Para el diseño de las interfaces gráficas se utilizó **SDL2**, así como distintos sub-sistemas de esta librería: **SDL_image** para el procesamiento de las texturas, **SDL_mixer** para los efectos de sonido, y **SDL_ttf** para las fuentes.
- Para el armado de los mapas utilizados *(que son basados en tiles)* utilizamos el editor libre **Tiled**.
- Para la edición gráfica del cliente se utilizó **Adobe Photoshop CS6** (diseño de ventanas de conexión, de log-in, de sign-up, escalado de distintos sprites, interfaces gráficas del juego en sí, etc.) en conjunto con el software libre **GIMP**.
- Para la generación de diagramas UML utilizamos la herramienta open-source **PlantUML**, en distintas implementaciones (algunos optaron por implementaciones *on-line*, mientras que otros decidieron utilizarla de manera *off-line*).

# 8. Conclusiones

Como grupo se encontró en este trabajo una gran oportunidad para trabajar en equipo y de manera coordinada, algo que muchas veces se intenta en otras materias pero pocas veces es realmente necesario para poder llegar a los objetivos. En este caso, esquematizar un **plan de desarrollo** se hizo indispensable. A su vez, se aprendieron nuevas formas de trabajo en conjunto, como por ejemplo el exhaustivo uso de la práctica conocida como **pair-programming**, que nos permitió minimizar los errores de diseño en esquemas nuevos a construir.

También se considera de gran utilidad la experiencia adquirida en la programación con **sockets TCP**, tema que para todos los integrantes fue completamente nuevo y muy interesante, así como la **programación multi-hilo**.

## 8.1. Conclusiones personales

En este espacio, cada miembro del equipo expone sus conclusiones personales:

- **Mauro Parafati:** yo me dediqué al armado del cliente y considero que los conocimientos adquiridos fueron muchos y muy variados, aprendiendo a armar vistas dinámicas, handlers de eventos de I/O, animaciones, sonidos, a reproducir música, a armar widgets desde cero, a integrar sprites dados en nuestro modelo, y muchas cosas más. A su vez, el armado de la máquina de estados del cliente me pareció particularmente interesante y muy divertida de hacer, ya que me dió mucha flexibilidad para las ventanas y para futuras extensiones de nuestro juego. Estoy muy contento con el resultado final y en lo personal este trabajo me ayudó mucho a aprender a trabajar en equipo.
- **Santiago Klein:** La elaboración del presente trabajo práctico me fue muy fructífera en múltiples aspectos, brindando una experiencia sumamente rica en lo que respecta al trabajo en equipo, la implementación de un proyecto de tamaño considerable, y la puesta en práctica de temas muy interesantes vistos en la materia. Mi dedicación fue exclusiva del lado del servidor y aporté al desarrollo del juego y su comunicación con los distintos clientes, integrando los distintos avances con los mismos.
- **Yuhong Huang:** al principio, estuve en la parte del cliente hasta que por el retiro de un compañero, tuve que pasarme a ayudar en la parte del servidor. Me dediqué desde ese momento entonces al armado del mismo en conjunto con otro compañero y considero que fue una buena oportunidad para poner en práctica los conocimientos que aprendimos a lo largo de la cursada. Participé en varias sesiones de **pair-programming** con mis compañeros, y aprendí mucho de sus buenas práticas de programación. También me resultó muy interesante la communicación entre cliente y servidor, así como el sistema de la persistencia, a través del cual aprendí mucho sobre manejo de archivos avanzado. Me siento muy feliz con el resultado final y personalmente siento que aprendí mucho en este trabajo.