**Final Programación II**

Alumno: Agustín Delger

Profesor: Jonathan L. Pepe

Materia: Programación II

Carrera: Desarrollo de videojuegos – IPCAC

Año: 2016

Como el enunciado lo indica, en el presente trabajo se implementó un ejemplo de arquitectura cliente-servidor. Cumpliendo con los requisitos pedidos, un cliente puede vincularse únicamente a un servidor, y solo si tiene conocimiento de la contraseña del mismo. Una vez vinculado, podrá enviarle tantas peticiones como desee, pero en caso de intentar enviar una sin haberse vinculado previamente, ocurrirá un error en tiempo de ejecución del programa. Por otro lado, un servidor puede tener vinculados muchos clientes.

Para tener control de los clientes vinculados, al momento de vincularse la clase Servidor solicita una referencia a una instancia de Cliente. El servidor tiene una lista de punteros a cliente, en la cual tendrá un puntero a cada cliente vinculado. Al tratarse de referencias a los clientes, para saber si un cliente está vinculado, solo hace falta comparar los punteros.

En cualquier momento, se le podrá solicitar al servidor que atienda peticiones durante una cierta cantidad de ciclos de reloj, priorizando las críticas de acuerdo con su nivel de prioridad. Para llevar a cabo esto, se precisó utilizar 3 instancias de Cola (clase que fue implementada por el docente) y se implementó, además, una cola de prioridad. Cuando se recibe una nueva petición, ésta es agregada a la cola de prioridad en caso de ser una petición crítica, o en la cola de peticiones no críticas en caso contrario. Las otras dos colas mencionadas son de uso auxiliar en el momento de atender peticiones.

Como indica la consigna, se deben atender primero las peticiones críticas de acuerdo a su prioridad, y posteriormente a las no críticas, de acuerdo con su orden de llegada; pero en caso de no haber suficiente tiempo para atender por completo una petición, no se lo hace y se procede a buscar la siguiente petición que sí pueda ser completada. Había dos formas de llevar a cabo esto. La forma no utilizada era agregarle un método al iterador tanto de la cola común como la de prioridad para poder eliminar el elemento apuntado. De esta forma, se podía recorrer los elementos buscando la próxima petición que pudiese ser atendida por completo, atenderla y luego eliminarla.

La segunda forma de implementar esto, y la utilizada en el trabajo, es utilizar dos colas auxiliares. Al momento de atender una nueva petición, se la desencola y se consulta si hay suficiente tiempo para atenderla. Si lo hay, se lo hace y se continúa con la siguiente. Pero en caso de no haber suficiente tiempo, se encola la petición en una u otra cola auxiliar, dependiendo de si se trata de una petición crítica o no crítica. Luego se repite la operación desencolando la próxima petición y nuevamente consultado si hay tiempo suficiente para atender. Se repite hasta encontrar una petición que cumpla lo buscado, o hasta que ya no haya más peticiones en las colas. Una vez llegado a este punto, solo resta reestablecer las colas. Para hacer esto, se desencolan todas las peticiones críticas de la cola auxiliar y se las encola nuevamente en la cola de prioridad. Es importante aclarar aquí que al encolarlas, éstas volverán a su posición original debido a su prioridad. Distinto sucede con las peticiones comunes, que si se las desencola de la auxiliar y se las encola nuevamente en la cola original, estas irán al final, detrás de aquellas que llegaron posterior a éstas. Para evitar esto, primero se terminan de pasar todas las peticiones de la cola original a la cola auxiliar. De esta forma, el orden original se respetará dentro de esta última. Finalmente, se vuelven a pasar todos los elementos a la cola original, preservando el orden nuevamente. No es un detalle menor que estas operaciones tienen un costo considerable (ya que siempre se recorren todas las peticiones, y en algunos casos se las recorre hasta 3 veces), por lo tanto esta metodología es aplicable en este trabajo, donde la cantidad de peticiones es pequeña pero no funcionaría de forma eficiente en una aplicación real con miles de peticiones. Para un caso como éste, se hubiese implementado la metodología de iterador.

Se muestra a continuación un diagrama de clases del trabajo. Es importante aclarar que por falta de presupuesto, falta hacer referencia a ciertas clases que juegan un rol importante como las peticiones, o algunas con un rol más secundario como la lista. Esto se debe a que la herramienta para realizar el diagrama online no informaba que había una restricción en el tamaño del proyecto si no se compraba la versión paga de la misma. Esto fue informado en el momento en que ya no se podía extender más, y para entonces el 80% del diagrama ya se encontraba hecho.

