PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Actividad 14

2^o semestre 2017
9 de noviembre de 2017

Networking

Objetivos

- Conectar el archivo `client.py` al `server.py` mediante el uso de *sockets*, de tal manera en que se puedan efectuar las acciones requeridas.
- Utilizar los métodos de una librería externa.

Introducción

El fin de semestre se acerca y con ello tus ganas de procrastinar. Encontraste un código con la lógica del juego *Batalla ~~Nebil~~ Naval*, creado por *Almirante Bad Bunny*[1]. Quieres jugar con alguien más, por lo que decides crear la versión *online*, para jugar con uno de tus amigos que no tomó este entretenido curso.

Batalla Naval

Esta batalla se juega en parejas. Cada jugador tiene una flota de barcos y su objetivo es hundir los buques rivales. Además, cada jugador tiene una vista del tablero con dos mapas: uno en donde están ubicados sus barcos y los ataques que le han realizado, y un segundo donde se registran los tiros realizados al oponente. **No es posible ver los barcos del oponente**, pero es posible descubrir sus posiciones mediante los ataques.

Al inicio, los barcos deben ser asignados de forma aleatoria en el tablero. En cada turno, el jugador debe atacar a su oponente indicando una posición del tablero. El juego debe finalizar en el momento en que alguno de los dos participantes logra hundir todas las piezas enemigas.

El archivo `battleship.py` tiene toda la lógica del juego. Lo que ustedes deben realizar en esta actividad sólo es la conexión entre los jugadores, creando un servidor que utilice los métodos de la librería `battleship`.

Librería `battleship`

La librería `battleship` consiste de dos clases: `Battleship` y `Board`. Ambas son necesarias para el funcionamiento interno de la librería. Sin embargo, para esta actividad, **sólo requieren llamar a los métodos públicos** que expone `Battleship`. En particular, su constructor, `attack`, `view_from`, `game_over` y `get_winner`. Pueden leer la documentación (como *docstring*) de los métodos en el mismo archivo `battleship.py`. Además, se incluye un *Jupyter notebook* (el archivo `demo.ipynb`) con un *demo* que puede ser de gran utilidad para entender cómo funciona la librería.

[1] El conejo malo.

Estructura *servidor-clientes*

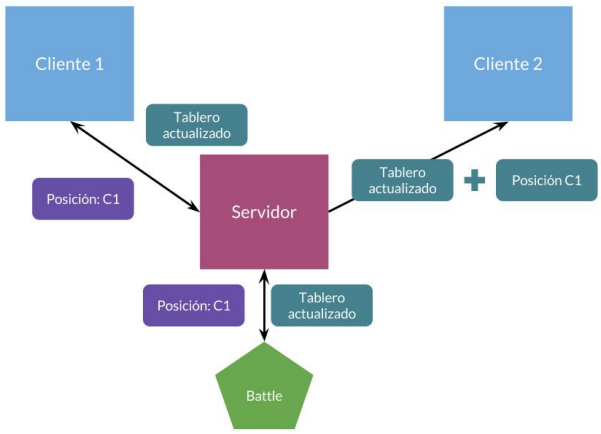
Cada jugador será cliente de un servidor que tendrá toda la información de la partida. Esto significa que el servidor es **el único** que puede interactuar directamente con la partida de *battleship*.

En cada turno los jugadores deben informarle al servidor la posición que quieren atacar. El servidor debe recibir esta información, realizar la acción en el juego y enviarle a ambos jugadores el estado de sus mapas después de la jugada. Además, debe informarle al jugador atacado cuál fue la posición atacada. Si uno de los jugadores quiere terminar antes el juego, debe enviar el mensaje `exit`. El servidor debe procesar esta información e informar al otro jugador de que ha ganado la partida. Lo mismo debe ocurrir si es que uno de los jugadores se desconecta sin avisar.

La forma en que se envían y reciben los mensajes **debe ser definida por ustedes**.

En la figura 1 se puede observar el flujo de información cuando el cliente 1 ataca. Primero, le informa al servidor que quiere atacar la posición `C1`, el servidor actualiza la batalla y esta le retorna el tablero actualizado. El servidor le envía el tablero actualizado a ambos clientes, y al cliente dos le informa cuál fue la posición atacada.

Figura 1: Diagrama del envío de información durante un turno.



Notas

- El primer jugador que se conecta es el primero en jugar.
- Para evitar problemas con los mensajes recibidos, puedes usar separadores entre mensajes, o *headers* que indiquen el largo del mensaje.
- Es una buena idea que el intercambio de mensajes entre el servidor y los clientes sea en forma de diccionarios. La llave del diccionario será el tipo de mensaje (e.g. `enviar_tablero`), mientras que el valor sería el contenido (e.g. el tablero en sí).

Requerimientos

- (5,6 pts) Conexión clientes-servidor
  - (0,8 pts) Se conectan correctamente ambos clientes al servidor y los reconoce.
  - (0,8 pts) Se inicia la partida solo si hay dos jugadores.

    - (0,8 pts) Durante el turno de un jugador, lo que realice el otro jugador no tiene efectos sobre el tablero.
    - (0,8 pts) Se envían y procesan correctamente las posiciones atacadas.
    - (0,8 pts) Se recibe correctamente los dos mapas.
    - (0,8 pts) Ambos jugadores reciben la información sobre el término del juego.
    - (0,8 pts) Maneja el caso de la desconexión o término del juego.
- (0,4 pts) Utiliza correctamente los métodos necesarios de la librería.

Entrega

- **Lugar:** En su repositorio de GitHub en la **carpeta** `Actividades/AC14/`
- **Hora:** 16:55
- Si está trabajando en pareja, basta con que un miembro suba la actividad. Si se suben actividades distintas, se corregirá una de las dos al azar.