Práctica - Bloque 1: TCP

Ítem 1: Diseño de un multiplayer online

Implementa (y acaba de diseñar) un **juego para 4 o más jugadores** que tenga sentido jugarlo online.

Dado que TCP funciona mejor en juegos que no utilizan tiempo real, el juego que diseñes no debe ser necesario jugarlo en tiempo real o el tiempo real no debe ser determinante.

En este caso:

- Se pide escoger entre una de las temáticas del PDF PosiblesJuegosAA3.
- La partida debe empezar una vez llegan todos los jugadores. Las condiciones de finalización del juego dependerán de las reglas que acabéis de definir.

El juego debe tener interfaz gráfica, aunque sea muy sencilla. **Y puede ser muy sencilla.**

Ítem 2: Funcionamiento/Reglas del juego y protocolo de comunicación

Describe cuáles son las reglas del juego:

1) Cuál es la situación inicial en el juego
2) Cómo evoluciona
3) Cuáles son las condiciones de finalización y quién es el ganador

Vamos a implementar toda esta funcionalidad utilizando las topologías client/server y peer-to-peer.

La topología client/server funciona de una forma concreta y hace que el protocolo de comunicación necesite intercambiar mensajes entre clientes y servidor. Esto genera un protocolo de comunicación ad-hoc a esta topología.

En la topología peer-to-peer sabemos que no hay servidor de juego y todos los nodos tienen su propia copia del estado del juego. Eso puede hacer que el protocolo de comunicación para conseguir la funcionalidad final cambie respecto a client/server.

El objetivo de este apartado es que estructures un protocolo de comunicación para cada caso.

Protocolo de mensajes para Client/Server

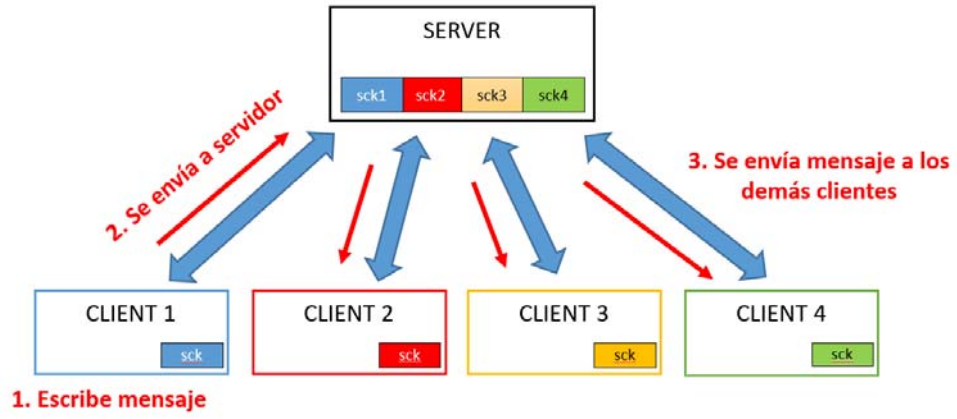
Estructura mensaje	Sentido	Acción asociada

Protocolo de mensajes para Peer-to-Peer

Estructura mensaje	Acción asociada

Cuando vayas implementando, estas listas irán cambiando. Procura tenerlas actualizadas. Te ayudarán.

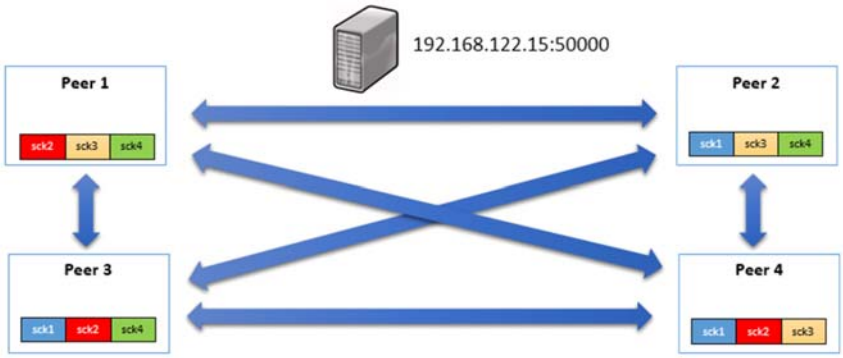
Ítem 3: Implementación en topología cliente/servidor



Recuerda:

1. El servidor no tiene parte gráfica. Pero es el que mantiene el estado del juego. Así que debe guardar la información de todos los jugadores. Recuerda que para esto se utilizan los ClientProxy.
2. El cliente tiene sólo la información del juego necesaria para la representación gráfica. El jugador envía los comandos a servidor y no puede ejecutarlos hasta que el servidor le "da permiso". Sólo si servidor lo considera válido, se ejecuta en este cliente y en todos los demás (ya que todos deben ver la misma versión del juego).
3. El servidor indica cuándo empieza la partida y asigna turnos (si los hay).
4. El servidor indica cuándo finaliza la partida y quién es el ganador.

Ítem 4: Implementación en topología peer-to-peer



Recuerda:

1. En esta topología tenemos un Bootstrap Server que nos ayuda sólo al inicio para que todos los nodos establezcan conexión.
2. En el momento que se dan cuenta de que ya han llegado los N jugadores necesarios, ya puede empezar la partida.
3. A partir de ahí, todos los nodos están al mismo nivel. Todos tienen una copia "verdadera" del mundo.
4. Todos los nodos pueden comunicarse con todos los demás.
5. Cada nodo valida sus propias acciones y da por buenas las acciones de los demás. (Podría añadirse alguna verificación extra, pero no es necesario para esta práctica).

Ítem 5: Juego en modo cliente/servidor con varias salas de juego

La implementación cliente/servidor debe permitir que el servidor mantenga varias partidas a la vez.

Concretamente, serán los propios jugadores lo que creen sus propias salas de juego o se unan a salas ya existentes.

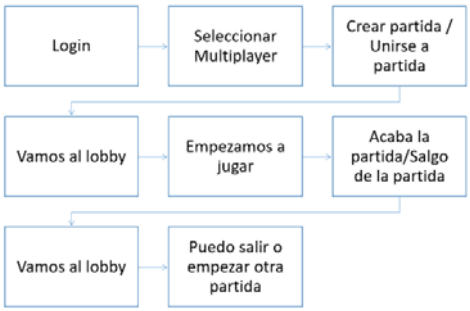
El jugador que quiera crear una sala de juego debe:

1. Darle un nombre que no exista para otra sala de juego.
2. Poder configurar al menos un parámetro de la partida. Por ejemplo: Tiempo de juego o tiempo de turno, tamaño del mapa, etc.

El jugador que quiere unirse a una sala puede:

1. Ver un listado de las salas disponibles en el que aparecerá, al menos, el nombre de la sala.
2. Hacer búsquedas. Si sólo quiero jugar en mapas "pequeños" debo poder buscar salas de juego en las que se haya elegido ese tipo de mapa. También se podría hacer un filtro por salas que no estuvieran llenas. Vosotros debéis hacer al menos un filtro de búsqueda.

En ambos casos, los jugadores deben poder comunicarse a través de un chat. Podrás elegir cuál es el lugar más adecuado para hacerlo.



En los apuntes de clase se analiza este flujo de navegación del jugador a través de las diferentes fases de creación / unión a partidas. Ten claro cuál va a ser el flujo para tus partidas.

Recuerda:

1. Cuando seleccionamos la sala de juego, no entramos directamente en la partida. Debemos ir a parar a un lobby en el que, como mínimo, nos podremos ver con el resto de jugadores
2. Cuando la partida acabe, el cliente no se acaba, se le debe dar la oportunidad de jugar otra partida.

Importante

A la hora de evaluar se considerará **muy importante**:

1. Que los protocolos de comunicación diseñados cubran todas las situaciones del juego
2. Que los protocolos de comunicación estén correctamente implementados
3. La implementación de la comunicación por red: Entre jugadores y servidor / entre nodos
4. Que estén controlados los errores de comunicación: Sobre todo a la hora de conectarse y desconectarse (acabar partida)

Entregable de la práctica → Protocolos

1. Un documento Word/Excel o similar en el que se detalle el protocolo utilizado para la topología cliente/servidor. El protocolo debe contemplar la creación/unión de/a partidas.
2. Otro documento Word/Excel o similar en el que se detalle el protocolo utilizado para la topología P2P. El protocolo debe contemplar la creación/unión de/a partidas.

Entregable de la práctica → Código

1. Nos aseguramos de que la solución compila en debug y en release.
2. Nos aseguramos de que el proyecto no utiliza paths absolutos para linkar las librerías.
3. Hacemos un clean
4. Eliminamos la carpeta oculta .vs
5. Hacemos un zip de la solución

Lo subimos todo al Classlife en este formato:

AA3_PracticaTCP_Nombre1Apellido1Nombre2Apellido2.zip

Rúbrica → En breve…