ESTUDIO RORSCHACH

Diseño de requerimientos y funciones de red de Last Bear Standing

Miguel Paniagua Muela
Miguel Córdoba Alonso
José María Ortiz García
José Roberto Martínez Gras
Jorge Puerto Esteban
Manuel Gómez Cámara

Índice

Diseño de la Arquitectura

La arquitectura de red sigue un modelo cliente/servidor. Tanto el cliente como el servidor actúan independientemente del otro y se comunican con mensajes. Es una arquitectura diseñada para el funcionamiento en LAN. Con un máximo de 4 jugadores por partida.
La red multijugador emplea Raknet, un motor de red en C++, para facilitar la comunicación entre el cliente y el servidor, ya que Raknet proporciona los protocolos UDP y TCP.

Requerimientos

En el Servidor

- Número de mapas
- Orden aleatorio de aparición de los mapas
- Número de players
- IPjugadores
- PuertoJugadores

En el Cliente

- ID del jugador
- IPserver
- PuertoServer
- Variables de estado de cliente
    - Posición
    - Cogiendo
    - Dirección
    - Mando
    - Muerto
    - Objeto Cogido
- Número jugadores en red
- ID de jugadores en red

Funciones

Servidor

Podemos dividir la función del servidor en 3 grandes bloques:

1. CONEXIÓN AL SERVIDOR

El servidor debe conocer siempre la cantidad de **jugadores en red**. Tras una conexión de un jugador al servidor, el servidor almacena la **IP** y el **puerto** del jugador correspondiente, se encarga de actualizar el número de jugadores en red, y envía a los clientes la ID del jugador entrante, de esta manera el jugador entrante obtiene una ID y el resto de jugadores advierten de un nuevo jugador y su ID.

2. INICIO DE PARTIDA

Una vez conectados los jugadores al servidor, el servidor se encarga de advertir del inicio de partida para los jugadores conectados. Para ello el servidor envía a los jugadores el mensaje correspondiente acompañado del **número de mapas** y su **orden aleatorio establecido**.

3. ACTUALIZACION EN BROADCAST

El servidor se encarga de flujo de mensajes de cada uno de los clientes o jugadores conectados. El servidor pues, recibe el mensaje de un determinado cliente, y en función del **puerto** e **IP** transmite este mensaje al resto de clientes conectados, permitiendo así la actualización en tiempo real de todos los clientes.

Cliente

En este caso podemos dividir la función del cliente en:

1. CONEXIÓN AL SERVIDOR

El cliente solicita una **IP** de conexión que almacena, y genera un puerto automáticamente de cuatro dígitos aleatorios. Una vez establecida la conexión recibe un mensaje de servidor y obtiene una ID de jugador, con la que podrá ser visto por el resto de jugadores, y las ID de los jugadores ya conectados, de esta manera obtiene el **número de jugadores en red** y sus **ID.**

2. INICIO DE PARTIDA

Una vez recibido el inicio de partida por parte de servidor, cliente actualiza al mundo (el propio juego), del **número de jugadores** en red, sus **ID**, **número de mapas** y **orden de aparición en la partida** y da paso al juego.

3. ACTUALIZACION EN BROADCAST

La manera en la que trabaja el cliente es por eventos y el juego lleva a cabo una simulación de estos eventos.

El cliente actualiza al resto de jugadores por eventos, esto es, el cliente solo envía el mensaje pertinente a través de un evento concreto. Por ejemplo, si un jugador decide saltar, cliente captará ese evento y enviará a servidor el mensaje correspondiente para notificar que el jugador saltó, el resto de clientes tras conocer esa información llevan a cabo una simulación *in game* del salto de ese jugador.

Esta forma de proceder puede generar pequeñas diferencias en los distintos clientes, por ello, cliente conoce todas las **variables de estado** de un jugador lo cual permite actualizar todo su estado cada 1-5 segundo/s al resto de clientes.