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

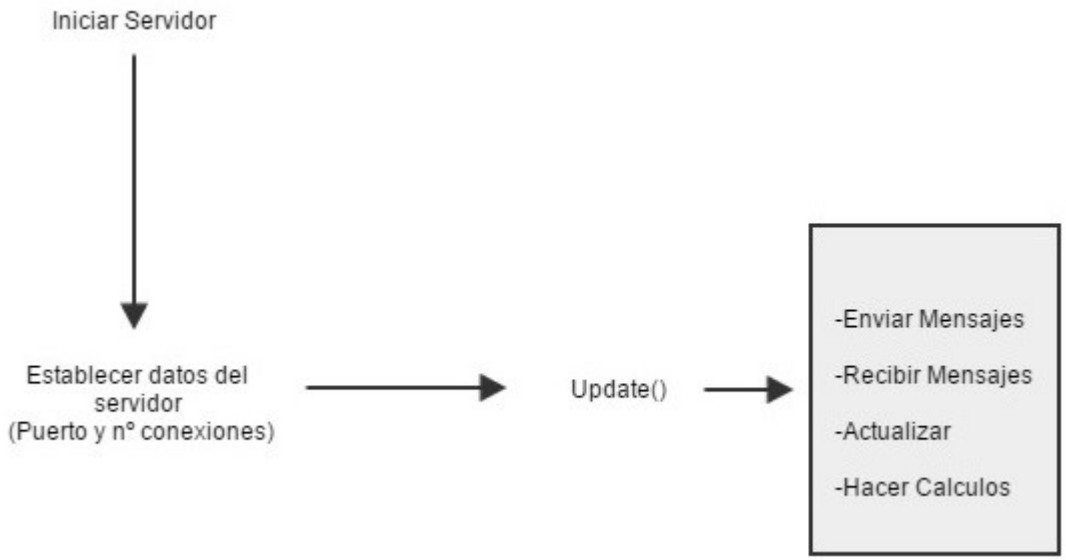
Índice

Diseño de la Arquitectura

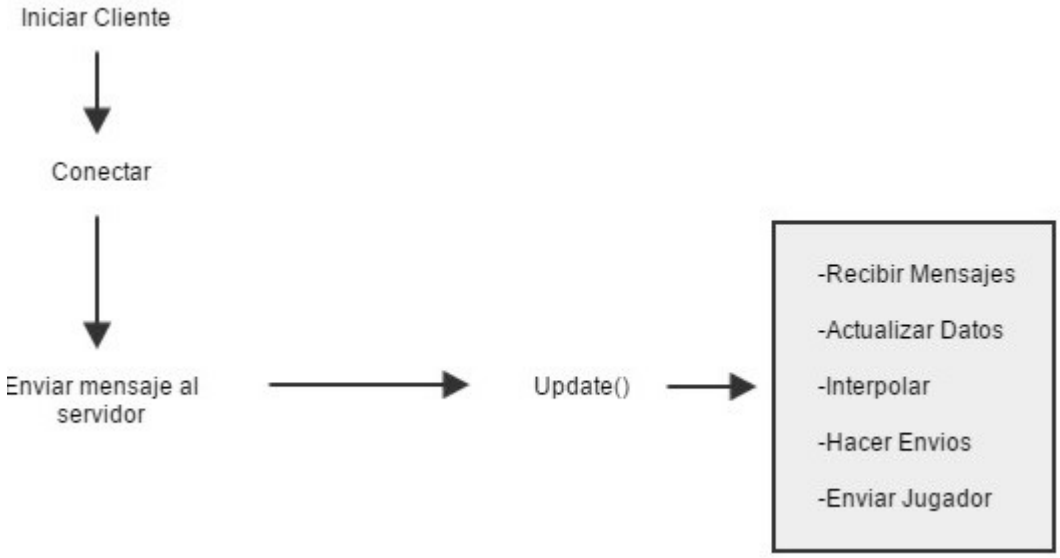
La arquitectura de red que vamos a utilizar es la cliente/servidor. Tanto el cliente como el servidor actuarán independientemente del otro y se comunicarán con mensajes. Es una arquitectura diseñada para el funcionamiento en LAN. Con un máximo de 4 jugadores por partida.
Para implementar la red multijugador se usaremos 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.

Funcionamiento de la red.

Servidor



Cliente



Cálculos a realizar

En el Servidor

Los cálculos que se aplicaran en el servidor será el de la IA, fuzzy logic, pathfinding, etc. Siempre que en la partida haya algún bot, ya que se utilizan para rellenar, siempre que falten participantes y la aparición y desaparición automática de las armas

En el Cliente

En el cliente se harán los cálculos propios para cada personaje, saltar, disparar, hacerse el muerto, coger objetos, etc. cada uno el suyo y enviarán un mensaje al servidor para que este transmita a los otros clientes estos datos, los cuales deberán actualizar los datos e interpolar las posiciones.

Mensajes

Servidor

Enviar

1. **WELCOME:** Se envía después de conectar a un cliente.
2. **UPDATE_PLAYERS:** Envía los datos de todos los jugadores a todos los clientes.
3. **UPDATE_BOT:** Envía los datos de los bots, siempre que hayan.
4. **UPDATE_BALAS:** Envía los datos de las balas disparadas.
5. **UPDATE_ARMAS:** Envía los datos de las armas dispersas por el mapa.
6. **UPDATE _ELEMENTOS:** Envía los datos de las entidades sin IA.
7. **SERVER_TO_CLIENT:** Envía mensaje del servidor a los jugadores.

Recibir

1. **ID_NEW_INCOMING_CONECTION:** Recibe una nueva conexión al servidor, por parte de un cliente y se le envía el mensaje WELCOME.
2. **ID_DISCONNECTION_NOTIFICATION:** Recibe si un cliente se ha desconectado y envía el mensaje SERVER_TO_CLIENT para informar a todos los clientes.
3. **UPDATE_PLAYERS:** Recibe los datos de un jugador.
4. **UPDATE_ BALAS:** Recibe los datos de una bala.
5. **UPDATE_ARMAS:** Recibe los datos de si se han cogido o tirado armas.
6. **UPDATE_ELEMENTOS:** Recibe los datos de si se está usando alguna de las entidades sin IA.

Cliente

Enviar

1. **ID_NEW_INCOMING_CONECTION:** Se envía cuando se quiere conectar al servidor.
2. **UPDATE_PLAYERS:** Envía los datos del jugador.
3. **UPDATE_BALAS:** Envía los datos de las balas disparadas.
4. **UPDATE_ARMAS:** Envía los cambios de las armas.
5. **UPDATE_ELEMENTOS:** Envía los cambios en las entidades sin IA.
6. **ID_DISCONECTION_NOTIFICATION:** Se envía cuando se cierra la aplicación y se desconecta.

Recibir

1. **WELCOME:** Recibe el mensaje de conexión.
2. **ID_CONECTION_REQUEST_ACCEPTED:** Recibe la aceptación de la conexión

3. **UPDATE_PLAYERS:** Recibe los datos de los otros jugadores
4. **UPDATE_BOT:** Recibe los datos de la IA
5. **UPDATE_BALAS:** Recibe los datos de las balas de otros jugadores
6. **UPDATE_ARMAS:** Recibe los datos de las armas del mapa
7. **UPDATE_ELEMENTOS:** Recibe los datos de las entidades sin IA
8. **SERVER_TO_CLIENT:** Recibe un mensaje del servidor