Proyecto
Lab21

Grupo
Dire Wolf Games

DISEÑO DE RED

Hito:
Fecha entrega: 23-12-2016
Versión: 1.0

Componentes:
- Aarón Colston Avellà Hiles
- Sergio Huertas Ferrández
- Eduardo Ibáñez Frutos
- Marina López Menárguez
- Rubén Moreno Mora
- Rafael Soler Follana

Contenido

1. Introducción

1.1. Propósito

Desarrollar un sistema de control para multijugador en red para aplicarlo en el Videojuego. Hemos elegido el motor de red RakNet para implementar este sistema, el cual posee licencia de uso BSD que permite su uso, modificación y distribución libre manteniendo el copyright de la empresa Oculus VR.

2. Descripción

2.1. Servidor.

La parte servidora utiliza los plugins de RakNet:

- *NatPunchthrough: maneja las conexiones entre máquinas aun estando detrás de Routers.*
- *CloudServer: Permite la comunicación entre máquinas y almacena los datos de cada uno de los clientes, permitiendo compartirla entre varios servidores.*

El servidor se ejecuta en modo consola creando un objeto RakPeer y lanza una conexión a través del puerto 61111 mediante el método Startup, permaneciendo a la escucha hasta que se presiona la tecla 'q'.

Administra las conexiones entrantes y la comunicación entre las máquinas de manera automática.

2.2. Cliente.

El cliente maneja la comunicación de RakNet a través de una fachada en la carpeta Net, que contiene las clases NetGame encargada de la comunicación con el Servidor y DrawableReplica que se encarga de enviar los cambios de la posición y rotación de un objeto a los demás clientes conectados.

*Para usar NetGame se debe iniciar con una llamada al método **open()**, el cual crea una instancia de RakPeer y una conexión para las comunicaciones entrantes mediante Startup (al igual que el servidor), además configura los siguientes plugins de RakNet:*

- *ReplicaManager3: necesario por DrawableReplica para la replicación de los datos de los objetos en las demás máquinas de manera automática.*
- *NatPunchthroughClient: conecta a un servidor Nat a través de Routers.*
- *CloudClient: conecta al CloudServer realizando las operaciones en envío y recepción de paquetes.*
- *FullyConnectedMesh2: conecta con las demás instancias de RakPeer buscando la que lleva más tiempo conectada que actuará de Servidor.*

Por último, open() llama al método Connect de RakPeer que realiza la conexión con el servidor.

2.3. Objetos replicables.

Se utiliza el objeto DrawableReplica para la replicación de los datos de posición, rotación y demás necesarios en los demás ordenadores. Los personajes controlados por los jugadores necesitan enviar periódicamente sus actualizaciones. Serán heredados de DrawableReplica para realizar el envío de datos. Ejemplo: clase Player, que en los demás equipos crea un objeto PlayerMate.

2.4. Estrategia.

Para los objetos que son manejados por los jugadores, los cambios en estos objetos son enviados periódicamente a los demás equipos. El periodo en el cual se envía es configurado mediante la clase ReplicaManager3DireW, que hereda de RakNet::ReplicaManager3, usando el método SetAutoSerializeInterval, que por defecto es 30ms. La clase Replica3 se encarga de leer los datos que se le pasan mediante el método Serialize() y solo envía si ha cambiado alguno.

La IA del juego es calculada en cada uno de los clientes y periódicamente se controla que todo vaya sincronizado en los clientes mediante el envío de mensajes de comprobación de estados de las entidades.

Se elige utilizar el sistema 'time warp' para la solución de problemas de lag así como incoherencias en los clientes.