

Documentación TelePong

Proyecto N1

Profesor:

Juan Carlos Montoya

Integrantes:

Juan José Muñoz

Pedro Cárdenas

Andres Felipe Rua

Fecha

15 octubre 2023

Enlace Github:

<https://github.com/JuanJmf01/pong-videogame>

Introducción

En este proyecto se realizó un video juego pong en línea para que se pudiese jugar con múltiples consolas unidas vía internet, el propósito.

de este fue aprender como por medio de sockets podríamos conectar un servidor en línea funcional.

Documentación.

1. Estructura y contenido

Definiciones y especificación de requerimientos

a) Definición general del proyecto de software: Este proyecto es la recreación del famoso juego “PONG” de 1972.

b) Especificación de requerimientos del proyecto:

**Cliente:** El cliente (PongClient) puede ser escrito en varios lenguajes que admitan sockets (Python, java, etc.)

**Servidor:** El servidor (PongServer) debe ser implementado completamente en lenguaje C

**Logger:** Implementar un “Logger” en el servidor para registrar peticiones, errores y respuestas.

Despliegue en la nube: Desplegar el servidor en un entorno en la nube utilizando AWS Academy.

**Documentacion**

Documentar el diseño y especificaciones del protocolo de comunicación, utilizando un diagrama UML u otras herramientas.

Proceso de compilación

Utilizar el compilador gcc para compilar el código

Generar un archivo Makefile para facilitar la compilación y gestión del código

Alcance

**Aprendizaje y Desarrollo**: Puede ser una plataforma para aprender y desarrollar habilidades de programación y diseño de juegos, ya que implica la implementación de la lógica del juego, la comunicación en red y la interfaz de usuario.

**Experiencia en la Nube**: El despliegue en un servidor en la nube, como AWS, brinda experiencia en la administración de servidores en un entorno real.

**Diversión y Competencia**: El juego Pong en línea proporciona una experiencia de diversión y competencia para los jugadores. Los jugadores pueden disfrutar de partidas emocionantes y desafiantes contra oponentes de todo el mundo.

Limitaciones

**Latencia y Sincronización**: La latencia de red puede afectar la sincronización y la experiencia del juego. La distancia entre los servidores y los jugadores puede influir en la latencia.

**Complejidad Técnica**: La implementación de un juego en línea conlleva una complejidad técnica, que incluye la comunicación en red, la gestión de servidores y la seguridad. Esto puede requerir un conocimiento profundo de programación y redes.

**Escalabilidad**: La escalabilidad puede ser un desafío si el juego debe manejar un gran número de jugadores simultáneos. A medida que aumenta el número de jugadores, se requieren servidores más potentes y estrategias de escalabilidad.

2. Requerimientos técnicos

Tener los siguientes requisitos instalados:

[GCC](https://gcc.gnu.org/) (para compilar el código)

Abrir una terminal y actualiza la lista de paquetes disponibles para asegurarte de que este actualizada:

sudo apt update

Instalar GCC y otras herramientas esenciales de desarrollo utilizando el siguiente comando

sudo apt install build-essential

Verifica la instalacion nuevamente usando el comando:

gcc --version

Para compilar el proyecto, utiliza el siguiente comando: gcc main.c manejarPelota/manejarPelota.c servidor/serverSocket.c variables/variablesCompartidas.c variables/constantes.c -o main.out

Con el comando ./main.out se inicializa el servidor y se crea un socket, donde se pueden unir dos clientes.

Con el comando python3 transporte.py inicializamos el juego como cliente.

c) Procedimientos de instalación y prueba

Especificaciones generales:

A modo breve de explicación de las funcionalidades, en game.py es donde se inicializa y se definen todas las variables del juego básicas, el tamaño de la pantalla, raquetas y pelota acá también se definen las teclas de movimiento de los jugadores y el socket de estas, también la pantalla donde se colocan el apodo los jugadores y el tablero de puntaje

En manejarPelota es donde tenemos desde el servidor la actualización constante de la pelota, sus físicas y como esta actúa cuando choca con la pared, con el jugador o cuando el jugador no le da a la bola, acá también se controla el delay que tiene la misma y la tasa de refresco del servidor para que sea más optimo.

En serverSocket se maneja ya toda la parte del servidor-cliente acá se maneja el control de las partidas y de los clientes que ingresan donde se maneja que por partida solo reciba un máximo de 2 clientes y si llega un tercero este no podría jugar porque haría falta de otro cliente para poder ingresar a la partida, esto se refleja en un ciclo donde por partida solo recibe 2 clientes y si llegan otros 2 clientes este ciclo creara otro hilo y unirá los dos clientes a este nuevo hilo, también se manejan los sockets del servidor y de los datos del juego.

En constantes se inicializan las variables ya declaradas en los clientes, pero ahora en el servidor para que ambos clientes observen la misma pantalla y en variablesCompartidas se inicializan las variables en común del cliente y del servidor.

Arriba están las especificaciones de prueba necesarias para la ejecución del programa, pero se colocará un resumen en esta sección.

1. Descargar la carpeta del github: <https://github.com/JuanJmf01/pong-videogame> ubicado en la rama nickname
2. Descargar los compiladores de C, gcc en tu consola
3. Compilar el proyecto con gcc main.c
4. Crear un socket de juego con ./main.out
5. Ingresar al juego con python3 transporte.py

3. Definición General del proyecto de software

● **Idea general**: La idea general del proyecto es crear un servidor funcional de Pong online

● **Objetivos**: Este proyecto tiene como objetivo principal el desarrollo de habilidades avanzadas en el ámbito de la programación en red y las aplicaciones concurrentes, al mismo tiempo que busca fomentar el diseño y la implementación de protocolos de comunicación

● **Usuarios**: Este proyecto tiene un alcance a los compañeros estudiantes de clase, como al profesor para poder realizar la debida calificación del funcionamiento de este.

**Requisitos**

● **Requisitos generales**:

* Desarrollar la aplicación Cliente y la aplicación Servidor.
* Diseño, especificación e implementación de protocolo
* Su servidor debe ser capaz de soportar múltiples clientes.

● **Requisitos funcionales:**

**Partidas Multijugador**: Los jugadores deben poder unirse a partidas multijugador y jugar contra otros jugadores en tiempo real.

**Movimiento de Raquetas**: Los jugadores deben poder controlar las raquetas y moverlas arriba y abajo para golpear la pelota.

**Detección de Colisiones**: El juego debe detectar colisiones entre la pelota y las raquetas, así como entre la pelota y las paredes.

**Puntaje y Marcadores**: El juego debe llevar un registro del puntaje de los jugadores y mostrarlo en tiempo real.

**Manejo de Errores**: El sistema debe manejar errores y excepciones de manera adecuada, como la pérdida de conexión de un jugador.

**Registro de Actividad y Logs**: Debe llevarse un registro de actividad que incluya eventos importantes, como peticiones entrantes, errores y eventos de juego.

**Seguridad de Datos**: La comunicación entre el cliente y el servidor debe ser segura para evitar trampas y garantizar la integridad de los datos del juego.

4. Alcances del sistema

Alcances

**Experiencia Multijugador en Tiempo Real**: El proyecto permitirá a los jugadores experimentar partidas multijugador en tiempo real, lo que agrega una dimensión social y competitiva al juego.

**Registro y Visualización de Puntajes**: El juego registrará y mostrará el puntaje de los jugadores en tiempo real, lo que proporciona un incentivo adicional para competir.

**Interacción en Línea**: Los jugadores podrán comunicarse en línea a través de un sistema de chat, lo que mejora la interacción social y la experiencia de juego.

**Implementación de un Protocolo Personalizado**: El proyecto implica la creación y la implementación de un protocolo de comunicación personalizado entre el cliente y el servidor, lo que brinda la oportunidad de diseñar una solución adaptada a las necesidades del juego.

**Despliegue en la Nube**: El servidor se desplegará en un entorno en la nube, lo que brinda experiencia en la administración de servidores en un entorno real.

Limitaciones

**Complejidad Técnica**: La implementación de un juego en línea conlleva una complejidad técnica significativa. La gestión de la latencia, la sincronización, la seguridad y la escalabilidad son aspectos desafiantes que deben abordarse.

**Latencia de Red**: La latencia de red puede influir en la calidad de la experiencia de juego. A medida que la distancia entre el servidor y los jugadores aumenta, la latencia se vuelve más notable.

Especificaciones de procedimientos

**Entorno de desarrollo utilizado**: Visual Estudio Code

**Plataformas**: AWS y Linux

5. Planificación

Primero se organizó el servidor para poder mandar mensajes entre clientes, mensajes básicos, hola y de ese tipo, seguido de se creó un pong con turtle, esto para organizar la parte de la interfaz de los clientes, consiguiente a esto, realizamos cambios en los clientes para poder manejar las raquetas y jugar con el servidor, juntamos los clientes al servidor y ya podíamos controlar las raquetas, luego hicimos el movimiento de la pelota en el servidor para poder jugar apropiadamente, finalizamos esta parte del juego inicial montándolo todo a AWS y para finalizar optimizamos los clientes, la tasa de refresco y el encolamiento para solucionar algunos problemas y realizamos el Logger.

Procedimientos de instalación y prueba**:**

● Requisitos no funcionales

**Tiempo de Respuesta**: El sistema debe responder a las acciones de los jugadores en tiempo real, minimizando el retardo perceptible.

**Escalabilidad**: Debe ser capaz de manejar un creciente número de jugadores sin degradación del rendimiento.

**Protección contra Trampas**: El sistema debe estar diseñado para detectar y prevenir trampas y comportamientos no autorizados que puedan afectar la equidad del juego.

**Seguridad de la Comunicación**: La comunicación entre el cliente y el servidor debe ser segura para proteger los datos del juego y la privacidad de los jugadores.

**Alta Disponibilidad**: El servidor debe estar disponible la mayor parte del tiempo para permitir que los jugadores disfruten del juego en cualquier momento

**Interfaz de Usuario Intuitiva**: La interfaz de usuario debe ser fácil de entender y navegar para los jugadores.

**Accesibilidad**: Debe ser accesible para personas con discapacidades, cuando sea posible.

6. Arquitectura

La información sobre la arquitectura debería incluir como mínimo:

● **Descripción jerárquica**: El sistema en principio solo tiene dos componentes principales, el cliente y el servidor, dentro de el cual estan los subcomponentes de estos mismos, donde el principal de todos sera el servidor ya que es el que controla todo lo que sucede en el juego.

● **Diagrama de Clases**:Diagrama

Descripción generada automáticamente

**Descripcion General**

**Servidor**: Este modulo que contiene los 4 siguientes submodulos, es el servidor, lo que este debe de realizar constantemente es el control, actualizacion y comunicación tanto de los clientes, como de la pelota y el que tiene todas las variables del juego donde los clientes pueden estar en constante comunicacion

**serverSocket**: El serverSocket es el nucleo central de el servidor lo que hace principalmente es gestionar las conexiones de clientes, crear y administrar hilos para cada una de las partidas de pong que se creen y manejar la comunicación entre el servidor y los clientes de juego. Donde este para funcionar realiza:

Define la función crearHiloPartida que se utiliza para crear un hilo para una nueva partida de Pong. Verifica si hay jugadores disponibles para una partida y crea un hilo para administrar esa partida.

Define la función definirHiloPartida que se encarga de asignar jugadores a una partida en función de su posición en el arreglo de clientes. Luego, llama a crearHiloPartida para crear un hilo de partida.

Define la función startGame que se utiliza para iniciar un juego y enviar mensajes a los clientes. Esto implica la asignación de jugadores a partidas, el envío de datos del juego y la comunicación con los clientes.

Define la función newClient para manejar la llegada de nuevos clientes. Registra la información del cliente, como su dirección IP, puerto y nombre, en un arreglo de clientes.

Define la función conectTwoPlayers que se encarga de administrar las conexiones entrantes. Utiliza la función select para verificar si hay actividad en los sockets y luego procesa los mensajes entrantes de los clientes.

Finalmente, define la función defineSocket que configura el socket del servidor para escuchar conexiones entrantes en una dirección IP y puerto específicos.

**manejarPelota**: Este modulo se encarga de la física de la pelota en el juego, calculando su posición y dirección así como de la comunicación con los clientes para mantenerlos actualizados sobre la posición de la pelota y el puntaje del juego. Las funciones que esta define para poder realizar de forma correcta el modulo son:

Define la función definirPosicionBola, que calcula la nueva posición de la pelota en función de su posición actual, velocidad y colisiones con los bordes de la pantalla y las raquetas de los jugadores.

Dentro de definirPosicionBola, se verifica si la pelota ha colisionado con los bordes superior o inferior de la pantalla y se invierte su dirección si es necesario. También se verifica si ha colisionado con las raquetas de los jugadores y se invierte la dirección si es el caso. Además, se maneja el reinicio del juego y el aumento de puntajes cuando la pelota atraviesa los bordes laterales de la pantalla.

Define la función enviarPosicionBola, que se ejecuta en un hilo separado. Esta función actualiza continuamente la posición de la pelota y envía información sobre la posición y el puntaje del juego a los clientes. Utiliza dos búferes, uno para cada jugador, ya que la pantalla puede estar reflejada para el jugador opuesto. Los datos se envían a través de sockets a los clientes.

El ciclo while (1) en enviarPosicionBola controla la actualización continua de la posición de la pelota y la comunicación con los clientes. Se utiliza una pausa de 66,666 microsegundos (aproximadamente 15 actualizaciones por segundo) para controlar la velocidad de actualización del juego

**Constantes**: En este modulo es donde se configuran las constantes graficas del juego, pantalla, raquetas, el radio de la bola, velocidades y la escalabilidad del pong, este módulo es la configuración básica inicial del juego. Para realizar esto:

Define las constantes ADRESS\_IP y PORT, que representan la dirección IP y el puerto en los que se ejecutará el servidor del juego.

Configura la variable MAX\_CLIENTS para definir el número máximo de clientes que el servidor puede admitir. Esta variable se utiliza para determinar cuántos clientes pueden conectarse al servidor simultáneamente.

Define MAX\_PARTIDAS como la mitad de MAX\_CLIENTS, lo que sugiere que cada partida de Pong implica dos jugadores.

Establece varias constantes relacionadas con la configuración de la ventana del juego, como el alto y ancho de la ventana, las dimensiones de la raqueta y la pelota, y las velocidades de movimiento.

La función inicializarConstantes se encarga de calcular valores específicos en función de las constantes anteriores y la variable nivelEscala. Esta función se utiliza para ajustar las dimensiones del juego según la escala elegida.

**variablesConstantes**: Este modulo prepara las constantes y variables necesarias para gestionar el servidor del juego Pong, incluyendo la información del juego y los detalles de los clientes que se conectan.

En este modulo se definen las siguiente variable y estructuras con su funcionalidad:

server\_socket que almacena el socket del servidor. Este socket se utiliza para aceptar conexiones entrantes de los clientes.

Declara un array de hilos llamado hilos\_partidas, con capacidad para 10 hilos. Estos hilos se utilizan para manejar partidas de Pong en paralelo.

Define una estructura de datos llamada DatosDeJuego, representada como un array de 10 elementos, que almacena información relevante sobre el estado del juego, como la posición de la pelota, las raquetas y el puntaje.

Y otra una estructura de datos llamada ClientInfo, representada como un array de 20 elementos, que almacena información sobre los clientes que se conectan al servidor. Esto incluye detalles como el socket del cliente, el puerto, la dirección IP y el nombre del cliente.

**Cliente:** El modulo cliente es el que contiene el juego y el transporte, este modulo se encarga a grandes rasgos de la interfaz grafica y la logica del juego, asi mismo como de la comunicación de los clientes ya que este es el transporte de los mismos clientes con el servidor y luego con el otro cliente.

**Game**: Este modulo maneja la interfaz gráfica y la lógica del juego, tales como el movimiento de paletas, la colisión de la bola y el puntaje, también la interacción con otro cliente a través de una funcionalidad de red para sincronizar el estado del juego entre dos jugadores.

Este modulo implementa unas funciones que hacen que todo funcione de la mejor manera las cuales son:

actualizar\_juego(data) que se utiliza para recibir datos del juego y actualizar la interfaz gráfica en consecuencia. Estos datos pueden incluir la posición de la bola, las paletas y los puntajes.

juego\_inicial que muestra la ventana inicial en la que los jugadores deben ingresar su nombre antes de jugar. Esta función también maneja la interacción con el botón "Play".

juego\_grafica que inicia el bucle principal del juego. En este bucle, se maneja el movimiento de las paletas, la lógica del juego y se actualiza la interfaz gráfica.

Inicia dos hilos en paralelo: uno para la interfaz gráfica (hilo\_grafico) y otro para la recepción y envío de mensajes (hilo\_cliente) utilizando la biblioteca threading.

**Transporte**: Este modulo es el encargado de manejar la conexión entre el servidor y el estado del juego, aca se mandan y se reciben mensajes del servidor para manejar el estado del juego. Para realizar esto este modulo realiza lo siguiente:

Crea un socket de cliente (client\_socket) configurado para comunicarse con el servidor mediante el protocolo UDP (User Datagram Protocol).

Define una función llamada enviaConfirmacionServidor(nombre) que se utiliza para enviar un mensaje de confirmación al servidor con el nombre del jugador.

Definir una función llamada enviar\_posicion\_al\_servidor(message) que envía la posición del jugador al servidor. Esta función es utilizada para enviar mensajes que no son de confirmación al servidor.

Definir una función llamada recibir\_enviar\_mms() que se ejecuta en un bucle infinito y se encarga de recibir y enviar mensajes al servidor utilizando el protocolo UDP. La función utiliza la función select.select() para monitorear múltiples sockets y verificar si hay datos disponibles para leer en el socket del cliente.

Cuando se reciben datos del servidor (sock == client\_socket), se decodifican y se manejan según el tipo de mensaje. Los mensajes que comienzan con "GAME:" se almacenan en una cola de mensajes para su posterior procesamiento en el juego. Otros mensajes se envían al módulo de juego (game.actualizar\_juego(data)) para su procesamiento.

**● Dependencias externas**:

Este software utiliza las siguientes librerias de C y de python

#include <stdio.h>

#include <string.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

import socket

import select

import sys

import threading

import turtle

import queue

import pygame

Estas librerias nos sirvieron para realizar un pong mas optimizado y apropiado para la objetividad del proyecto, se escogieron estas por la utilidad que nos presentaron tanto para realizar la interfaz grafica del juego como el servidor.

Los lenguajes de programacion que utilizamos fueron C en el servidor porque con C podemos utilizar de forma mas optima el servidor por la robustez que tiene este lenguaje y python para la parte de clientes por el manejo de los integrantes y la cantidad de librerias que nos ayudaban con el uso de sockets y con el juego en si.

Se utilizaron estas tecnologias porque en nuestra fase inicial de consulta y preguntas al profesor se recomendaron estas ya que eran muy robustas y tenian gran cantidad de utilidades necesarias para el proyecto.

Conclusiones

Se pudo concluir que: El proyecto era bastante retador por el poco uso y conocimiento que teniamos con respecto a los sockets y los protocolos, sin embargo.

Logramos aprender el debido uso de un servidor en linea y como programar este mismo, identificamos el diferente uso de los sockets y como estos funcionan para permitir la comunicación en tiempo real entre el servidor y los clientes.

La elección de protocolos y técnicas adecuadas, en UDP es crucial para garantizar una comunicación eficiente y receptiva en un juego en línea.

Aprendimos la escalabilidad que pueden tener los servidores y como estos trabajan en base a esta misma, y comprendimos aun mas la importancia de la

sincronizacion y latencia en estos proyectos.

A nivel de UDP aprendimos a manejarlo y utilizar su buena comunicacion multicast que estos tienen, aunque llegamos a presentar problemas con la perdida de paquetes, logramos optimizar estas y gestionarla de forma que no afectase la experiencia del usuario como nos pudo afectar en un momento y aprendimos mas de como estos protocolos suelen afectar un proyecto en base cuando no se optimiza completamente.

A nivel de encolamiento pudimos concluir que aunque logramos que no hubiera un encolamiento en el juego y tuviera falencias cuando habian mas de 2 clientes, comprendimos el importante uso de una buena optimizacion a nivel del servidor en la nube donde todo estuviese correctamente organizado y no presentara estos problemas en un futuro.

Se pudo concluir que la implementación de la comunicación en tiempo real entre clientes y servidores es un desafío técnico importante. La elección de protocolos y técnicas adecuadas para transmitir datos de manera eficiente y segura fue esencial para el éxito del juego en línea.

Aprendimos que en un juego en línea, la optimización del rendimiento es crucial. Esto implica evitar repeticiones innecesarias de cálculos, minimizar la sobrecarga de la red y garantizar que el código sea lo más eficiente posible.

Referencias

<https://www.ponggame.org>

<https://www.programacion.com.py/noticias/sockets-en-c-parte-i-linux>

<https://www.tamos.com/htmlhelp/tg/int/es/udp_upstream_and_downstream_lo.htm>

<https://www.redeszone.net/tutoriales/internet/tcp-udp-caracteristicas-uso-diferencias/>

<https://stackoverflow.com/questions/4713582/simple-online-pong-game-network-synchronization>

<https://barcelonageeks.com/aplicacion-simple-de-cliente-servidor-en-c/>

<https://es.stackoverflow.com/questions/571255/codigo-de-c-funciona-diferente-en-servidor>