PyDakarDeath

Comprensión de arquitectura cliente-servidor con el fin de comprender el uso de sockets

Armando Fallas Garro 2019226675

Kevin Arias Sancho 2019209063

Kevin Calderón Esquivel 2019157479

Instituto Tecnológico de Costa Rica

Contacto: [armandofg3937@gmail.com](mailto:armandofg3937@gmail.com)

ariassanchok@gmail.com

[kevinec2000@gmail.com](mailto:kevinec2000@gmail.com)

Índice

[Introducción 3](#_Toc8926072)

[Descripción del problema 4](#_Toc8926073)

[Diagrama UML 5](#_Toc8926074)

[Análisis de resultados 6](#_Toc8926075)

[Bitácora 7](#_Toc8926076)

[Estadística 8](#_Toc8926077)

[Conclusión Personal 9](#_Toc8926078)

# Introducción

Para la comprensión del desarrollo en arquitectura cliente-servidor es necesario comprender el uso de sockets.

El proyecto está basado en el juego PyDakarDeath en el cual hay mucha destrucción, para llevar a cabo el trabajo se debe trabajar con programación orientada a objetos, archivos CSV, JSON, pero antes de esto es importante importar la librería pygame y tkinter ya que el juego necesita de ambas para que funcione, también dar una buena investigación al uso de sockets ya que estos son los encargados en que el programa sirve con un servidor hecho en este trabajo.

El juego debería poder jugarse ya sea en una misma computadora o un par de estas, por lo cual en el juego tanto el jugador1 como el jugador2 se podrán ver aunque no compartan la misma computadora.

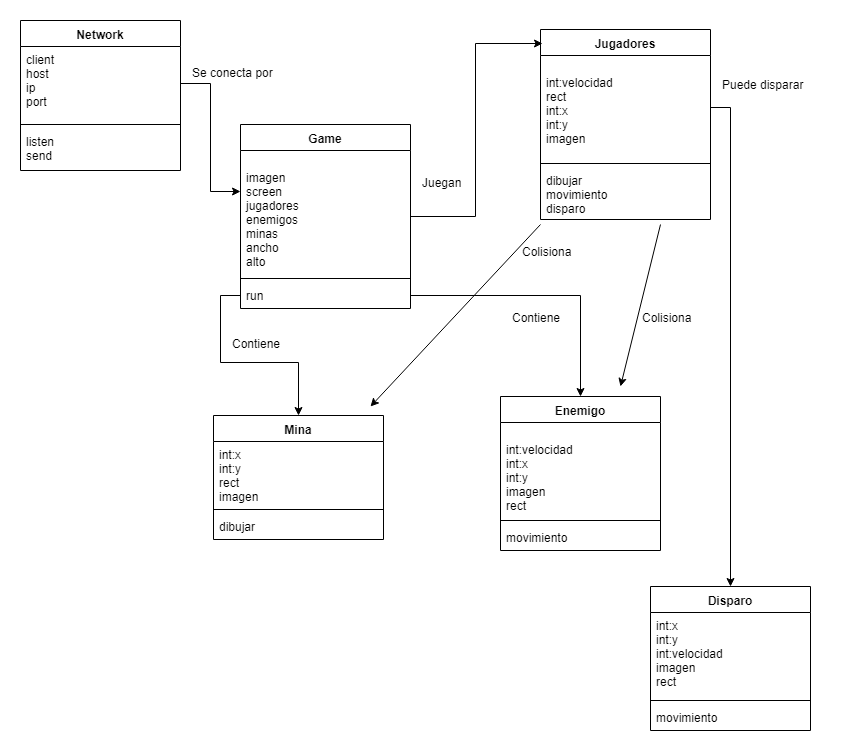
# 

# Descripción del problema

El juego debe contar con servidor, y 2 o más clientes, además de esto los jugadores experimentarán una cercanía a un juego online, esto quiere decir que ambos se podrán ver en la pantalla, además de esto aparecerán obstáculos mientras avance el tiempo en el juego, cada 2 minutos se subirá el nivel del juego, tanto el carro de los jugadores y los enemigos podrán disparar, además de que tendrá un escenario (estático).

Se podrá registrar el nombre, ver puntajes, guardar partida, y pausar partida. Y por último contará con sonidos de explosiones, aceleración y frenado.

# Diagrama UML



# Análisis de resultados

# Bitácora

|  |  |  |
| --- | --- | --- |
| Actividades | Tiempo Utilizado | Descripción |
| Investigación de sockets | 8 horas | Se trató de ver videos, investigación de guías, ayuda del tutor… |
| Implementación de servidor-cliente | 20 horas | Tras mucha prueba y error, se pudo lograr que el servidor y el cliente funcionaran. |
| Creación del UML | 40 minutos | Se le pidió ayuda al tutor para comprender como hacer el UML hasta que se logró. |
| Imágenes, canción, sonidos en el formato que se necesita… | 20 minutos | Se buscó en internet y se tuvo que buscar un convertidor para la música. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Estadística

# Conclusión Personal