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.

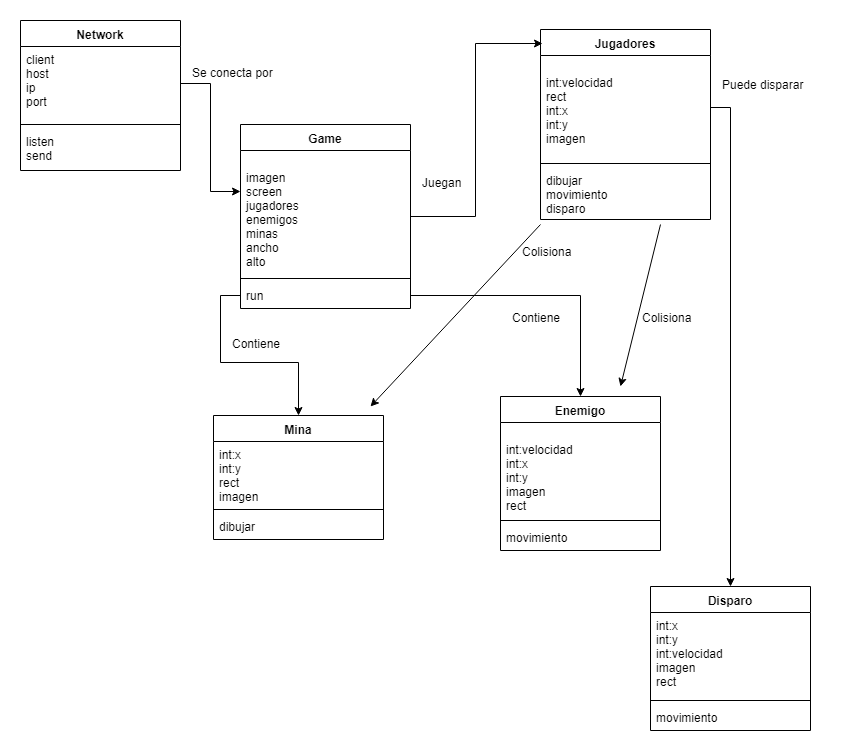
# 

# 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

Los requerimientos básicos para correr el programa son: descargar pygame, este link tiene las instrucciones para hacerlo https://skellykiernan.wordpress.com/2015/01/04/python-pygame-install/#install\_pygame, además de esto tener Python https://www.python.org/downloads/ que se puede descargar en el segundo enlace es muy fácil y rápido, y por último se llama la librería tkinter.

Además debes importar ciertas librerías que están especificadas en el código (Game.py).

No se pudo lograr un menú al pausar el juego, el guardar la partida tampoco se pudo lograr, además no se pudieron crear diferentes niveles para los jugadores.

# 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. |
| Se creó el fondo, se añadió los autos (jugadores) y bombas al juego. | 6 horas | Las minas aparecerán cada cierto tiempo y en diferentes lugares. |
| Se le añadió movimientos al carro, y que la imagen del carro rote dependiendo a donde vaya.(además también se incorporaron los disparos) | 8 horas | Se tuvo que investigar sobre como rotar o mover una imagen para lograr que en la programación se viera como es debido. |
| Creación de la ventana principal y algunos botones. | 30 minutos | Se creó el botón para jugar, salir, etc… |
| Se creó la pausa del juego y reanudar la misma. | 4 horas | Se tuvo que buscar cómo poder guardar la información que se tenía anteriormente. |
| Se le añadieron los sonidos al juego. | 1 hora | Se tomó como base el primer proyecto. |
| Todo lo hecho en el juego tratar de que lo corra junto al servidor y se vea en ambas computadoras. | 8 horas | Se tuvo que investigar mucho, además de pedir ayuda al tutor y explicaciones en clase. |

# 

# Estadística

|  |  |
| --- | --- |
| Análisis de requerimientos | 2 Horas |
| Diseño de la aplicación y diagrama de clases | 1 Horas |
| Investigación de funciones | 10 Horas |
| Programación | 60 Horas |
| Documentación interna | 1 Hora |
| Pruebas | 3 Horas |
| Elaboración del documento | 1 Horas |
| Total | 78 Horas |

# 

# Conclusión Personal

Con base en la investigación realizada, se aprendió a poder crear un servidor que reciba información y la envié a sus clientes, aunque dicha tarea no fue fácil de hacer, en algunos campos se hizo un repaso visto en el primer proyecto(programación orientada a objetos), el poder enviar información de un lugar a otro fue un enorme reto, que se pudo lograr en cierta medida, el trabajo en equipo y la coordinación fue mucho más exigente, administrar el tiempo de una forma eficiente era necesario, se podría decir que nos estamos preparando para lo que vendrá después.