Proyecto PaCE man

Área de Ingeniería en Computadores

Instituto Tecnológico de Costa Rica

CE3104 - Lenguajes, Compiladores e Intérpretes

Estudiantes

Allan Josué Calderón Quirós - 2018114634
Ronny Josué Santamaria Vargas - 2018109283
Antony Fabián Fallas Elizondo - 2018178906

Profesor

Marco Rivera Meneses

I Semestre 2020

Manual de Usuario

Requerimientos de Hardware y Software:

Las características recomendadas de hardware son :
-Procesador: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, 2304 Mhz, 4 procesadores principales, 8 procesadores lógicos
-Memoria física instalada (RAM): 8.00 GB
-Espacio de almacenamiento: Al menos 1GB de ram para todos los programas necesarios
Tipo de sistema : x64-based PC
**requisitos basados en las características del hardware con las que se realizó el proyecto

Las características recomendadas de software son :
SO: Microsoft Windows 10 Home
Java Development Kit 1.8 (explicado con detalle más adelante)
Java Runtime Environment (explicado con detalle más adelante)
IntelliJ Idea (explicado con detalle más adelante)
Visual Studio Code
Compilador de C para windows (gcc)
**requisitos basados en las características del software con las que se realizó el proyecto

Juego:

El programa es una implementación del conocido juego “PaCE-man” el cual trabajará en dos partes un servidor y un cliente, siendo el cliente el juego.

Este juego se basa en el principio del juego original, donde el usuario tiene control del PaCE-man utilizando las flechas del tablero de su computador y debe ir recorriendo el tablero comiéndose los puntos que se encuentran en él para poder ganar, a su vez debe escapar de los fantasmas ya que si choca con ellos perderá una vida, el jugador solo cuenta con tres. En el juego además de los puntos originales contará con frutas y power ups.

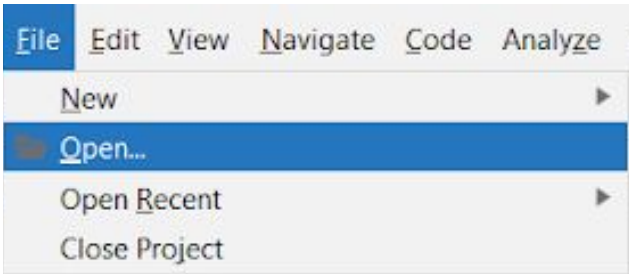
Las frutas le dan al jugador puntos extras y en el momento que el jugador se coma un power up tendrá un tiempo limitado en el cual puede comerse a los fantasmas, los cuales se tornan de un color azul durante ese lapso de tiempo.

A la hora de inciar el juego se muestra una pantalla con el nombre del juego y un botón que dice “New Game”, el cual se debe seleccionar y automáticamente inicia el juego. Al iniciar el juego el tablero solo contará con PaCE-man y los puntos originales. En el transcurso del juego se irán mostrando los demás objetos(fantasmas, powerups, frutas); dichos objetos se muestran en pantalla en el momento que el el servidor lo desee.

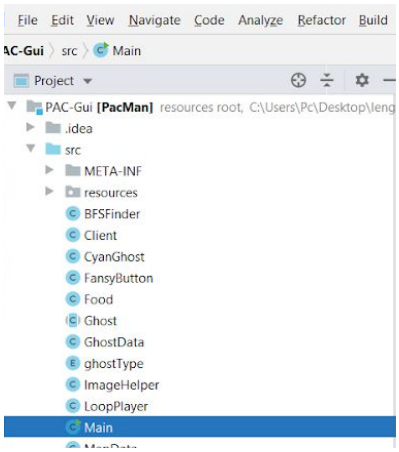
Los fantasmas contarán con distintas velocidades, que serán seleccionadas el servidor a la hora de crearlos.Las frutas y los power ups se crean en puntos definidos.

Para poder jugar, se debe ingresar a la carpeta del juego, llamada PAC-Gui y ahí se encontrará un archivo ejecutable llamado PaCE-Guía.Simplemente se debe ejecutar el archivo con doble click para empezar a jugar.
En caso de que archivo ejecutable no se encuentre se debe ingresar desde el código, para ello se debe tener instalado la aplicación de Intellij (ide en el cual se programó el juego), el cual se puede descargar desde la su sitio web: https://www.jetbrains.com/es-es/idea/download/#section=windows.
Además, se deberá instalar el JDK, el cual se puede obtener desde https://www.oracle.com/technetwork/es/java/javase/downloads/index.html.
y el JRE , el cual se puede descargar desde:
https://www.oracle.com/java/technologies/javase-jre8-downloads.html

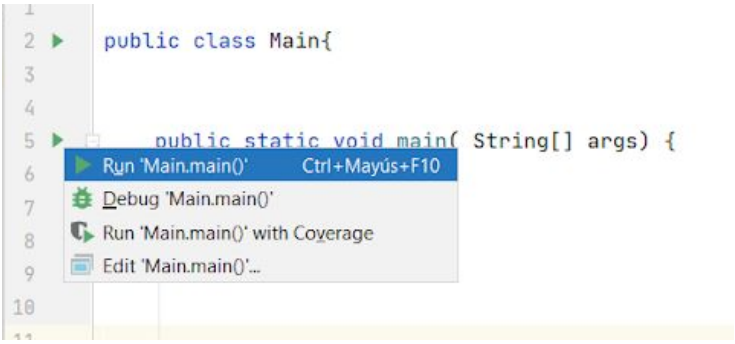
Una vez se tenga la aplicación instalada sólo se debe ingresar al juego. Para ello, se selecciona la pestaña “File”, luego se presiona “open” y se selecciona la carpeta en la cual se encuentra el juego.



Una vez que cargue aparecerán tres carpetas, una de ellas se llama PaCE-Man. Se debe ingresar a ella, luego se desplegarán otras carpetas, se debe ingresar a la de nombre PAC-Gui, ahí se encontrará una archivo llamado “Main”.



Posteriormente, se debe dar click derecho sobre . Aparecerá una flecha verde a la izquierda del archivo, la cual al presionar mostrará la opción “Run" ,desplegará una ventana con el nombre del juego y un botón que dice “New Game”. El juego inicia al presionar este botón.



Para jugar, PaCE-man deberá comerse todos los puntos amarillos del tablero sin ser tocado por los fantasmas, los cuales tratarán de encerrarlo.
Además de los puntos amarillos, PaCE-man podrá comer los power-up en forma de fruta, los cuales le darán puntos extra.

Una vez que el PaCE-man se ha comido todos los puntos, sube de nivel. El juego continúa hasta que PaCE-man pierda todas sus vidas.

Administrador del juego:

Cuando se ejecuta el servidor, se inicia con un texto que dice “Ingrese comando:”, después del texto se encuentra el espacio en el que se pueden ingresar las instrucciones que recibirá el juego. La estructura que presenta la instrucción es la siguiente: (Comando) (Identificador) (Complemento).

Comando: este se refiere a lo que se desea que haga el servidor, en caso de que se escriba el comando “create”, se dará la posibilidad de generar un enemigo o un potenciador en el juego. En caso de que la entrada sea “setSpeed” brindara la opción de cambiar la velocidad de un fantasma que se elegirá posteriormente en el identificador. En el caso en que se ingrese “exit” el servidor dejará de escuchar y terminará su ejecución. Si ninguno de los casos anteriormente descritos se cumple este retornara un mensaje de error de comando y volverá a escuchar la entrada de texto del terminal.
Identificador: el identificador es la letra del elemento que se quiere crear o modificar, en caso en el cual el comando sea “create” y el identificador sea “G” se creará un nuevo fantasma. Si por otro lado identificador es “P” va a generar una pastilla que proporciona la habilidad a Paceman para comerse a los fantasmas. Si el identificador es “F”, el servidor va a generar una fruta dentro del tablero de juego. Si el identificador no coincide con ninguno de los caracteres descritos, retornará un mensaje de error referente al identificador.
Complemento: Como su nombre lo indica este ayuda a complementar la información que se proporciona en las entradas mencionadas anteriormente. Para el caso en el que se desee generar un fantasma, basta con que se agregue como complemento cual fantasma se quiere agregar haciendo que complemento sea: “B” si se quiere generar a Blinky el fantasma rojo, “Y” si se quiere generar a Inky el fantasma azul, “P” si se quiere generar a Pinky el fantasma rosa y “C” si se quiere generar a Clyde el fantasma naranja. Si lo que se desea es generar una fruta la información que debería tener el complemento es el valor del puntaje que tendrá la fruta dentro del juego. En el caso en que se quiera crear una pastilla, el complemento debe ser la ubicación que tendrá en el tablero (coordenada en X y coordenada en Y). Si no se cumple con las condiciones necesarias se mostrará un mensaje de error referente al complemento.

1.Documentación

1.1. Descripción de las estructuras de datos desarrolladas:

1.1.1.Estructuras:

Una estructura(struct) en el lenguaje de programación C es una declaración de tipo de datos compuestos que define una lista agrupada físicamente de variables bajo un nombre en un bloque de memoria, lo que permite acceder a las diferentes variables mediante un único puntero.

En nuestro proyecto se aplicaron las estructuras en:

1.1.1.1 Almacenamiento de variables:

Se utilizaron para almacenar los valores de las constantes utilizadas por el administrador y por los sockets. Esto con el objetivo de poder cambiar su valor de una manera sencilla si fuera necesario. Algunas de estas variables son el puerto al que se conectan los sockets, la cantidad de bytes que el socket puede enviar y recibir por defecto y declaraciones de la longitud de mensajes fijos, tales como el mensaje de error que retorna el server si no se logra comunicar con el cliente de manera exitosa.

1.1.1.2.Coords:

Se creó con el fin de almacenar la posición de los personajes (paceman y los fantasmas) y de los potenciadores del juego como: las píldoras que permiten a pacman comerse a los fantasmas o las frutas.

1.1.1.3.PowerUps:

Esta estructura contiene la posición (Struct coords), en la que se va ubicar, puntos que proporciona al jugador, si el modo del paceman cambia (de ataque a escape o viceversa) y nombre este último ayuda a identificar si se crea una fruta o una píldora. En caso que la estructura sea una fruta tendrá una posición fija y si es una píldora de poder esta habilitará el modo de ataque de PaCEman.

1.1.1.4.PaCEman:

Se implementó un struct con el fin de mantener los valores importantes referentes a paceman, por ejemplo la posición (Struct coords) en la que se crea paceman, el modo en el que se encuentran (ataque o escape), cantidad de vidas y de puntos.

1.1.1.5.Ghost:

Se implementó un struct con el fin de mantener los valores importantes referentes a los fantasmas, por ejemplo la posición (Struct coords) en la que se crea el fantasma, el modo en el que se encuentran (ataque a paceman o escape), velocidad que tienen los fantasmas y el color característico del fantasma.

1.1.2.ArrayList:

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma dinámica.

En nuestro proyecto se utilizaron ArrayList en :

1.1.2.1.Tablero:
Se definió una matriz que almacena los valores del tablero, dependiendo de si en una posición de dicha matriz se encuentra un punto o fruta que PaCEman puede comer, un fantasma o PaCEman

1.1.2.2. Envío de datos en el cliente:
Se utilizaron ArrayList pequeños para enviar cambios a alguna posición.

1.2. Descripción detallada de los algoritmos desarrollados:

1.2.1.readLine:

Esta función se encarga de recibir el mensaje de entrada desde el terminal, divide el mensaje tomando como carácter divisor los espacios, tomando solo las instrucciones y los caracteres definidos para la ejecución de una función. Las funciones definidas son create que se encarga de crear un fantasma, píldora o fruta en el tablero de juego y a esta se accede mediante el comando “create”, setSpeed que cambia la velocidad de un fantasma, se accede usando “setSpeed” como comando y “exit” que se encarga de acabar la ejecución de la lógica del servidor. En el caso que no se cumpla con las condiciones descritas anteriormente se mostrará un mensaje indicando cual es el error.

1.2.2.setSpeedFunction:

Para acceder a esta función se debe ingresar “setSpeed (Fantasma al que se le debe cambiar la velocidad) (Nuevo valor de velocidad)”, esta se encarga de establecer un valor de velocidad diferente al fantasma seleccionado (“B”: Blinky, “I”: Inky, “P”: Pinky y “C”: Clyde). En el caso que no se cumpla con las condiciones descritas anteriormente se mostrará un mensaje indicando cual es el error.

1.2.3.createFunction:

Esta función crea en el tablero el objeto deseado ya sea un fantasma, una fruta o una píldora de poder, para acceder a esta función se debe introducir el comando “create (Identificador) (Característica)”. En caso de que el identificador sea “G” se creará un fantasma y la característica se refiere a cual fantasma (“B”: Blinky, “I”: Inky, “P”: Pinky y “C”: Clyde). En el caso en el que el identificador sea “P” se creará una píldora en el tablero y la característica es la posición que tomará en el tablero. Si el identificador es “F” se creará una fruta y la característica son los puntos que tiene la fruta. En el caso que no se cumpla con las condiciones descritas anteriormente se mostrará un mensaje indicando cual es el error.

1.2.4.parseSocketMessage:

Es la función que se encarga de recibir e interpretar el mensaje proveniente del servidor. Recibe una línea con caracteres definidos previamente para ejecutar una instrucción, por ejemplo el carácter “G” para referirse a cuando paceman se come un fantasma, sucedido de la letra que identifica al fantasma que se comio (“B”: Blinky, “I”: Inky, “P”: Pinky y “C”: Clyde) o “P” para pasar los puntos totales con los que cuenta paceman, este toma el mensaje entrante, divide el mensaje en cada espacio que este contenga, toma cada una de las partes en orden de izquierda a derecha donde el primer carácter se refiere a una instrucción (“G” o “P”) y el segundo al complemento de la instrucción anteriormente descrita.

1.2.5.Send_to_client:

Esta función se encarga de enviar mensajes desde el server hacia el cliente. La función recibe un mensaje al ser invocada, además de algunos parámetros necesarios para definir el tamaño de las cadenas de caracteres y el puerto al que se conectará el server. Posteriormente procede a iniciar una conexión por sockets, la cual escucha y pausa su ejecución hasta que le llega un mensaje desde el cliente. Una vez que envió el mensaje al cliente y recibió su mensaje, cierra la conexión de sockets y retorna un char*[] con el mensaje del cliente

1.2.6.Send_to_server:

Esta función envía mensajes desde el cliente hacia el servidor. Recibe de entrada el mensaje que se enviará al servidor y retorna el mensaje del servidor al cliente. Esta función incluye una declaración try-catch que maneja las excepciones en caso de que el server no esté escuchando o haya un error en la conexión. Esta función inicializa los sockets, envía un mensaje hacia el server por medio de los mismos, posteriormente lee el mensaje que el server dejó en el socket, cierra la conexión de sockets y por último retorna el mensaje del server. En caso de que la conexión con el server no sea posible, retorna un mensaje de error.

1.2.7.CrearFantasma:

Esta función se encarga de crear los fantasmas, recibirá como parámetro dos valores definidos como enteros los cuales corresponden; el primero al fantasma que se quiere crear(los cuales son valores del 1 al 4) y el segundo será la velocidad con la cual se moverá en el tablero(los cuales son valores del 1 al 9).

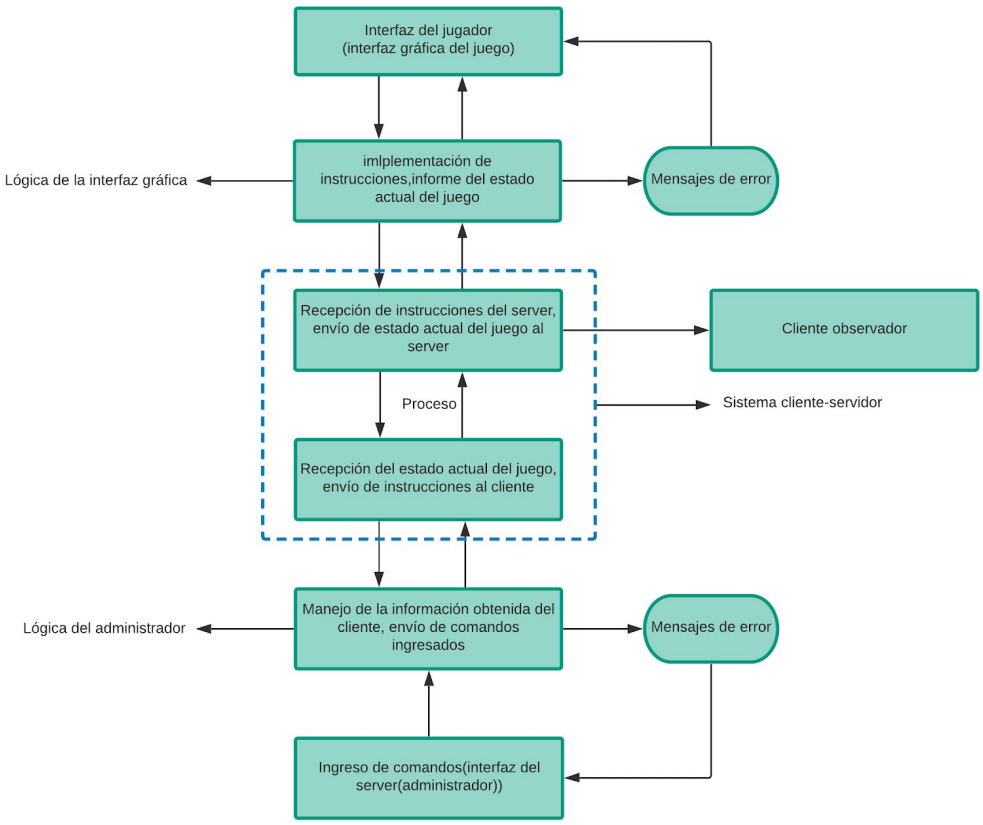
1.2.8.CollisionTest:

Esta función se encarga de verificar las colisión con un fantasma, primero verifica si existe una colisión comparando los posicion del PaCEman con la del fantasma, luego verifica si el jugador tenía el poder del power ups activado, en caso de ser cierto sumará al puntaje del jugador y enviará un mensaje al servidor. En caso de que no tenga el poder, verificará con cuántas vidas cuenta, si el número es diferente 1, volverá al Pacman a su posición inicial y le restara una, en caso de que su vida sea igual a 1, pausara el juego y mostrará un mensaje en pantalla diciendo que perdió.

1.2.9.Update:

Esta función se encarga de verificar cada vez que el jugador se come un punto, una fruta o un power up. En caso de que corresponda a una fruta o un punto sumará cierto valor al puntaje total del jugador según corresponda y si el objeto que se comió es un powerup le dará al jugador la opción de comerse a los fantasmas y sumar puntos cuando lo hace, pero esto solo es por un periodo corto de tiempo.

Diagrama de arquitectura:



1.3. Problemas sin solución:

1.3.1.El archivo Logic.c, el cual permite al administrador enviarle comandos al juego por medio de sockets, al ser ejecutado por primera vez , no permite la conexión entre el cliente y el servidor. Se desconoce porque se causa este problema, aunque se cree que proviene del antivirus o del firewall de windows, ya que el tanto el antivirus como el firewall lanzan advertencias luego de que el ejecutable recién compilado se ejecuta por primera vez.

1.3.2.El juego presenta un delay bastante grave(de varios segundos) debido a la conexión por medio de sockets. Se probaron varios códigos de cliente para intentar un mejor tiempo de ejecución, pero no se tuvo éxito. También se probaron varios códigos del lado del server, sin embargo, el tiempo tampoco mejoró. Se cree que es improbable que este fuera el problema, ya que el tiempo de ejecución no cambiaba incluso cuando el cliente no se conectaba con el server.
También se intentó mantener el server y el cliente siempre escuchando permanentemente(es decir, no cerrar la conexión). Se intentó abordar desde esta perspectiva de varias formas, tales como dejar el server escuchando pero cerrar la conexión del cliente, cerrar y abrir la conexion del server varias veces y no cerrar la del cliente, o no cerrar ninguna conexión. También se intentó incluir las llamadas tanto al cliente como al server en ciclos while(true).Lamentablemente ninguna de estas formas de abordar el problema funcionó. En cualquier intento, o bien el programa indicaba mensajes de error y se detenía(“se caía”), o la conexión entre el cliente y el servidor fallaba.
El problema parece provenir de la creación del socket del lado del cliente, parece que esta operación toma varios segundos. Al parecer la solución del mismo queda en un bajo nivel, actuando con el sistema operativo y con las bibliotecas de Java, por lo que lamentablemente parece quedar fuera de lo que se puede solucionar en un corto plazo.

1.3.3.En ocasiones y dependiendo del mensaje, al mensaje enviado desde el cliente hacia el servidor se obtienen algunos símbolos extraños al final de la cadena.Por la forma de manejar los mensajes desde el server, esto no representa un problema para la ejecución del programa.Se creé que esto se debe a que el cliente en Java está enviando datos de tipo String, mientras que el servidor en C debe recibir datos tipo char*. Esto probablemente causa una colisión de tipos de datos en el socket.

1.3.4.-Cuando el jugador pierde una vida, lógicamente se crea de nuevo en la su posición inicial como debe ser, pero gráficamente sucede que se crea una posición

más a derecha, es decir, si el jugador pierde una vida mientras se movía en las coordenadas “X”, para las siguiente vida se moverá gráficamente sobre las coordenadas “X” como debe ser pero a la hora de moverse en las coordenadas “Y” se moverá un espacio más a la izquierda de lo que debería y viceversa si a la hora de perder una vida lo hacia cuando se movía sobre las coordenadas “Y”. Esto solo sucede gráficamente, en la parte interna de la logicamente si se mueve como debe de ser.

1.4. Plan de Actividades realizadas por estudiante:

Encargado	**Tiempo estimado**	**Actividad**	**Fecha de entrega**
Allan Calderón	20 h	Implementación de los sockets, parseo de las instrucciones y adaptación a la lógica del juego	04/08
Ronny Santamaría	20 h	Implementación de los algoritmos de los fantasmas y el envío de instrucciones del server	04/08
Fabian Fallas	20 h	Implementación de la interfaz gráfica	04/08
Fabian Fallas-Allan Calderón-Ronny Santamaría	10 h	Debug, implementación y corrección de errores , manejo de instrucciones por parte del servidor y mensajes por parte del cliente	07/08

1.5. Problemas encontrados:

-Para la comunicación entre cliente y servidor, se utilizó en primera instancia el método DataOutputStream, de la clase java.io.DataOutputStream para enviar mensajes desde el cliente hacia el servidor. Sin embargo, al utilizar este método lo que el server recibía era impredecible. Durante la ejecución, al server le llegaba el mensaje cortado, con los caracteres desordenados, caracteres extraños, letras y números que no eran parte del mensaje original, el server recibía varios mensajes o la conexión fallaba. Debido a esto, se cambió por el método PrintWriter, de la clase java.io.PrintWriter para enviar el mensaje. Este método envía el mensaje correctamente, sin embargo en ocasiones se concatenan algunos símbolos extraños al final de la cadena. Por la forma de manejar

los mensajes desde el server, esto no representa un problema para la ejecución del programa.

1.6. Conclusiones y Recomendaciones del proyecto:

1.6.1:Conclusiones:

1.6.1.1.-El correcto manejo de una conexión de sockets (considerando los posibles escenarios, incluso cuando la conexión no sea posible) puede ralentizar el juego, debido al proceso en el cual el cliente debe intentar conectarse a un server, enviar un mensaje, recibir un mensaje y retorna el mensaje, o en caso de que la conexión no sea posible retornar un error.

1.6.1.2.-El presente proyecto fue desarrollado en los lenguajes de programación C y Java, lo cual genera una mayor inversión de tiempo en la resolución del problema, la sola implementación de ambos en un mismo proyecto ha provocado que las estrategias de resolución planteadas sean mucho más extensas y específicas que si solo se utilizara uno de los lenguajes. Sin tomar en cuenta lo anterior ya es un proyecto con requerimientos difíciles de conseguir, debido a que presentan los algoritmos que utilizan los fantasmas son tan distintos y extensos ya que evalúan el estado del juego y la mejor solución para un momento dado. La idea original fue implementar a cada uno de los fantasmas, con el mismo comportamiento que en el juego clásico de Pac-Man, pero esto si bien es cierto es alcanzable, requiere de un planteo más eficiente y detallado de los conceptos que presentan, esto para hacer uso de los algoritmos de una forma óptima. Debido a que los algoritmos de los fantasmas son extensos y requieren mucho procesamiento y de procesos extensos se optó por la implementación de los mismos en el mismo lenguaje en el que se desarrolló la interfaz, además de la razón ya descrita anteriormente el manejo de la comunicación de los sockets también fue un factor influyente, debido a la limitada capacidad de transferencia de los mismos, provocaría que el programa sea más extenso con el fin de optimizar los procesos y el tiempo para transmitir todos los datos al cliente aumente de manera considerable.

1.6.2.Recomendaciones:

1.6.2.1.-Optimizar el algoritmo de los sockets para tener un tiempo de envío de datos menor. Para esto, se pueden intentar utilizar otros algoritmos de sockets con otro protocolo (en este proyecto se utilizó TCP), revisar las bibliotecas y el mismo SO para encontrar fallos, o mantener una conexión constante con el server (nunca cerrar la conexión). En este proyecto se intentó implementar sin éxito esta última opción.

1.6.2.2.-Implementar los algoritmos de los fantasmas en el servidor, esto con el fin de que el código desarrollado para la interfaz se encargue solo de la verificación de funciones básicas, tales como graficación de elementos, recibir y enviar datos.
1.6.2.3.-Crear un mensaje de envío más compacto en cuanto a la cantidad de caracteres enviados, pero con la mayor cantidad de información posible para evitar una saturación en los sockets.

1.7. Bibliografía consultada en todo el proyecto:

Structures in C - GeeksforGeeks. (2020). Retrieved 22 July 2020, from https://www.geeksforgeeks.org/structures-c/

Pac-Man Patterns — Ghost Movement (Strategy Pattern) - DEV. (2020). Retrieved 24 July 2020, from https://dev.to/code2bits/pac-man-patterns--ghost-movement-strategy-pattern-1k1a

c, h., Shawi, B., Shawi, B., Bode, J., Larson, F., & Dodd, C. (2020). how to convert from char* to char[] in c. Retrieved 25 July 2020, from https://stackoverflow.com/questions/2074116/how-to-convert-from-char-to-char-in-c

C?, H., Galkin, A., & Rankin, D. (2020). How to convert an int to string in C?. Retrieved 27 July 2020, from https://stackoverflow.com/questions/8257714/how-to-convert-an-int-to-string-in-c/23840699

How to split a string in C/C++, Python and Java? - GeeksforGeeks. (2020). Retrieved 27 July 2020, from https://www.geeksforgeeks.org/how-to-split-a-string-in-cc-python-and-java/

"Complete Winsock Server Code - Win32 apps", *Docs.microsoft.com*, 2020. [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/winsock/complete-server-code. [Accessed: 04- Aug- 2020].

"Socket Programming in Java", *Www2.ic.uff.br*, 2020. [Online]. Available: http://www2.ic.uff.br/~michael/kr1999/2-application/2_06-sockettcp.htm. [Accessed: 03- Aug- 2020].

N. Minh, "Java Socket Client Examples (TCP/IP)", *Codejava.net*, 2020. [Online]. Available: https://www.codejava.net/java-se/networking/java-socket-client-examples-tcp-ip. [Accessed: 03- Aug- 2020].

"Java - Networking - Tutorialspoint", *Tutorialspoint.com*, 2020. [Online]. Available: https://www.tutorialspoint.com/java/java_networking.htm. [Accessed: 07- Aug- 2020].

"Writing the Server Side of a Socket (The Java™ Tutorials > Custom Networking > All About Sockets)", *Docs.oracle.com*, 2020. [Online]. Available:

https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html. [Accessed: 04- Aug- 2020].
. What does this symbol mean in IntelliJ? (red circle on bottom-left corner of file name, E. Green and H. Palappetty, "What does this symbol mean in IntelliJ? (red circle on bottom-left corner of file name, with 'J' in it)", *Stack Overflow*, 2020. [Online]. Available: https://stackoverflow.com/questions/4904052/what-does-this-symbol-mean-in-intellij-red-circle-on-bottom-left-corner-of-fil. [Accessed: 05- Aug- 2020].
H. file?, M. Flaschen and N. Fetissov, "How to use a defined struct from another source file?", *Stack Overflow*, 2020. [Online]. Available: https://stackoverflow.com/questions/3041797/how-to-use-a-defined-struct-from-another-source-file. [Accessed: 04- Aug- 2020].
B. Shawi, J. Bode, F. Larson and C. Dodd, "how to convert from char* to char[] in c", *Stack Overflow*, 2020. [Online]. Available: https://stackoverflow.com/questions/2074116/how-to-convert-from-char-to-char-in-c. [Accessed: 04- Aug- 2020].
M. Neacsu, "Windows Sockets Streams", *Codeproject.com*, 2020. [Online]. Available: https://www.codeproject.com/Articles/5252621/Windows-Sockets-Streams. [Accessed: 04- Aug- 2020].
Programación de juegos - Detección de colisiones. (2020. [Online]. Available: http://edu4java.com/es/game/game6.html.[Accessed: 31- Jul- 2020].

Interfaz Gráfica GUI - Programación Básica JAVA. (2020). [Online]. Available: https://sites.google.com/site/programacionbasicajava/interfaz-grafica-gui.[Accessed: 30- Jul- 2020].

2.Bitácoras digitales

Allan Calderón

Fecha	**Tiempo**	**Actividad**
24/07	1.5 h	Lectura de los requerimientos del programa, organización y división del trabajo
31/07	1 h	Investigación inicial sobre implementación de los sockets

01/08	9 h	Implementación de sockets
02/08	5.5 h	Implementación de sockets
03/08	5 h	Implementación de sockets
04/08	0.5 h	Documentación
05/08	8 h	Integración del cliente del socket con el juego, integración del server del socket con el administrador del juego, debug(intento de solucionar el delay obtenido al integrar el cliente con el juego)
06/08	11.5 h	Debug(intento de solucionar el delay obtenido al integrar el cliente con el juego), intento de implementación de la clase observador , documentación
07/08	2 h	Documentación, pruebas finales, preparación para la revisión
Total	44 h	

Ronny Santamaria

Fecha	**Tiempo**	**Actividad**
22/07	1h 30min	División general de tareas para cada miembro del grupo.
23/07 - 24/07	5h	Investigación de los algoritmos de persecución que utilizan los fantasmas.
25/07	2h	Definición de las estructuras necesarias para la programación del juego.
26/07 - 30/07	14h	Creación de lógica para el manejo de la entrada de datos desde la terminal.
31/07	2h	Unión de estructuras con la lógica de lectura de entrada de “comandos”.
02/08	2h	Definición e implementación del mensaje que será enviado al cliente.
04/08	1h 30min	Implementación de la funcionalidad que se

		encargará del mensaje entrante desde el cliente.
05/08 - 06/08	13h	Pruebas al funcionamiento completo de la lógica del servidor y corrección de errores.
07/08	2h	Documentación, pruebas finales, preparación para la revisión.
Total	43 h	

Antony Fallas

Fecha	**Tiempo**	**Actividad**
23/07	1 h	Lectura de los requerimientos del programa,
25/07	4 h	Investigación sobre formas de hacer interfaces gráficas en java
27/08	8 h	Implementación de la primer venta y segunda ventana
30/08	6 h	Implementación de los fantasmas y del pacman
31/08	5h y 30 min	Investigación e implementación de colisiones entre objetos
03/08	7h	Corrección de errores en el código
04/08	8 h	Integraciones del juego con los sockets y deteccion y correcion de errores
5/08	8 h	Creación de función donde se crean los fantasmas,power ups y frutas con la indicación del servidor
06/08	8 h	Pruebas de código, corrección de errores. se agrega sonidos al juego y documentación
07/08	3 h	Debug, documentación.
Total	58.5 h	