Instituto Tecnológico de Costa Rica

Proyecto Estructurado
Pac-man

Lenguajes de programación
IC4700

Estudiantes:
María José Barquero Pérez
Te Chen Huang
Kendall Rodríguez Mora

Profesor: Eddy Ramírez

22 de Diciembre, 2020

1. Resumen ejecutivo

Para este proyecto se pidió desarrollar el juego Pac-man con algunas variantes respecto al juego original,donde este como lo conocemos va a tener al héroe y a los cuatro fantasmas, donde Inky va a moverse aleatoriamente y los tres restantes van a tener una inteligencia diferente, siempre van a tratar de perseguir a Pacman buscando el camino más corto, estos tres fantasmas van a tener esta función en común pero cada uno tiene características únicas que los diferencian.

Este proyecto fue desarrollado en C por lo que decidimos utilizar la biblioteca Allegro 5 ya que esta se utiliza para desarrollar videojuegos y trae muchas funciones que facilita por ejemplo mostrar en pantalla el mapa con imágenes y los diferentes personajes. Esta biblioteca nos ayudó a lograr reconocer las teclas para permitir mover el pacman, poder representar a cada fantasma con su imagen y para poder mostrar el mapa decidimos crear un ciclo donde si es una C significa un coco, la S una semilla, ˜ representa las paredes y los espacios en blanco es la cueva de los fantasmas, de este modo por medio del ciclo según cada símbolo se van a representar en el tablero. El pacman cuenta con 5 vidas y todos los personajes son hilos independientes.

Al igual que los proyectos anteriores se realizaron pruebas, pero estas son con diferente cantidad de semillas si estas están distribuidas de manera uniforme o aleatoria, también se muestran pruebas donde se puede ver la velocidad de los personajes, cuando pacman come una semilla los fantasmas cambian de color y escapan de pacman. Se puede concluir de estas pruebas que según la cantidad de semillas que se ponga se va a tener más oportunidades para pasar al siguiente nivel.

2. Introducción

En este proyecto se pretende desarrollar el juego Pac-man con algunas variaciones, en el lenguaje de programación C con ayuda de la biblioteca Allegro 5, con el propósito de reforzar nuestro conocimientos en este. A lo largo del documento se intenta expresar de la mejor manera como logramos resolver este juego, las diferentes funciones de esta biblioteca, como se puede jugar y ejecutar el archivo del proyecto.

Como todo proyecto tuvimos algunas complicaciones, pero logramos resolverlo de la mejor manera buscando siempre el acuerdo de los integrantes del grupo y desarrollar el algoritmo de la manera más eficiente.

El documento se encuentra dividido en varias secciones donde primero se encuentra el resumen ejecutivo con pequeños detalles de qué trata este proyecto y cómo lo resolvimos. Un marco teórico donde mencionamos la historia, antecedentes, bibliotecas y usos del lenguaje utilizado en este proyecto. Se hará una descripción detallada de las soluciones y los resultados realizados con pequeñas pruebas donde estos van a ser reflejados con diferentes capturas de pantalla donde muestra cómo se comporta el juego según los parámetros que se ingresen. Y por último nuestras conclusiones del proyecto y un pequeño comentario de cada uno de los integrantes del grupo explicando cómo fue su experiencia en este trabajo.

Para el proyecto se presentó la opción de poder mostrar el juego como nosotros quisiéramos, utilizando la herramienta que consideremos mejor para desarrollar el proyecto. Investigamos y entre las bibliotecas encontradas, como grupo nos llamó más la atención Allegro 5 que durante lo largo de este documento se va a comentar el porqué de esta decisión.

3. Marco teórico

3.1. Historia

Ken Thomson creó el lenguaje B en 1969 a partir del BCPL (Basic Combined Programming Language) de Martin Richard. Utilizó el lenguaje ensamblador y el B para producir las versiones iniciales del sistema operativo UNIX. El BCPL y el B eran lenguajes sin tipos, en los que las variables eran simples palabras en memoria. Dennis Ritchie de los Laboratorios Bell convirtió más tarde (en 1972) el B en C (reteniendo la mayor parte de la sintaxis de B), y escribió el primer compilador. Se implementó en PDP 11 de DEC. Su primera utilización fue la de lenguaje para reescribir el sistema operativo UNIX. Después, UNIX, sus herramientas y C han crecido de forma simultánea. En 1978, Kernighan y Ritchie escribieron un libro titulado «El lenguaje de programación C» que se convirtió en la definición del lenguaje durante casi una década. A partir de 1983, se pidió al comité ANSI X3J11 que estandarizara el lenguaje C. El resultado fue el ANSI C, adoptado como patrón en 1988. No se impone a ningún programador, pero dada su amplia aceptación, a cualquier programador de sistemas o creador de compiladores le resultaría económicamente poco práctico no satisfacer este patrón.[10]

Muchos programadores han denominado el C como «lenguaje de nivel medio». No es por su falta de potencia programadora, sino por su capacidad de acceder a las funciones de bajo nivel de un sistema. El código C funciona tan rápidamente como el código del lenguaje ensamblador.

El lenguaje de programación C es un lenguaje de propósito general con aplicaciones casi infinitas.

Aunque C se considera un lenguaje de alto nivel, está mucho más cerca del hardware que la mayoría de los otros lenguajes. Por esa razón, muchos lenguajes que son incluso "superiores.ª C usarán C para compilar (Python, por ejemplo). Algunos de los usos principales de C incluyen: [11]

- Sistemas Operativos.
- Compiladores e interpretes de lenguajes.
- Ensambladores.
- Drivers.
- Aplicaciones de Bases de datos.

La directiva #include: indica una directiva preprocesador, que va a incluir" las declaraciones de otro fichero en la compilación. Es decir, que nos permite agregar librerías o funciones que se encuentran en otros ficheros al programa.

#include stdio.h: Es el archivo de cabecera que contiene prototipos, macros, constantes, declaraciones de funciones y la definición de tipos para realizar operaciones de Entrada y Salida.

#include stdlib.h: Es el archivo de cabecera que contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.

#include pthread.h: Es una librería que cumple con los estándares POSIX que nos permite trabajar con varios hilos distintos de ejecución al mismo tiempo.

#include unistd.h: El archivo de encabezado que define diversas constantes simbólicas, tipos y funciones.

3.2. Allegro5

Allegro 5 es una biblioteca libre de código abierto para la programación de videojuegos desarrollada en lenguaje de programación C que cuenta con muchas funciones muy útiles para gráficos, manipulación de imágenes, dispositivos de entrada y entre otras. Por otro lado, esa librería es compatible con diferentes sistema operativos incluyendo los de móviles IOS y Android.

3.2.1. Instalación en Linux (18.04+)

Primero que todo, se debe agregar el PPA de Allegro: sudo add-apt-repository ppa:allegro/5.2

Luego, se usa el siguiente comando para realizar la instalación:
sudo apt-get install liballegro*5.2 liballegro*5-dev

3.2.2. Estructura de la biblioteca y sus complementos

Allegro 5 está compuesta por una biblioteca principal y varios complementos. Los complementos se agrupan y compilan al mismo tiempo que el núcleo, pero esos se guardan en bibliotecas diferentes. El núcleo no depende de los complementos.

El archivo de encabezado de la biblioteca principal es “allegro5 / allegro.h”. Los archivos de encabezado de complementos que se utilizaron en el proyecto son los siguientes:

allegro5 / allegro_image.h, allegro5 / allegro_font.h, etc.

Para poder usar cada complemento de Allegro 5 se debe inicializar primero “al_init”. Cada uno tiene su propia inicialización, por ejemplo en el caso de imagen se utilizará el siguiente comando:

al_init_image_addon

Se utiliza el comando al_create_display para se abre una ventana y devolverá allegro_display. Para poder limpiar la interfaz se usa al_clear_to_color.

3.2.3. Estructura de la biblioteca y sus complementos

En el caso de insertar una imagen, como se menciona anteriormente se debe inicializar primero que sería utilizar el comando al_init_image_addon. Luego, se usa al_load_bitmap que devuelve allegro_bitmap, es decir retorna el puntero al bitmap creado, si es null entonces el bitmap no fue agregado correctamente.

3.2.4. Colas de eventos y entrada

Las colas de eventos agregan los eventos de todas las fuentes, en este caso como entrada utilizamos el teclado y el mouse. Crear una cola de evento: al_create_event_queue

Colocar eventos

al_register_event_source(queue,al_get_keyboard_event_source());

al_register_event_source(queue,al_get_display_event_source(disp));

El IDE utilizado fue Sublime Text, que como hemos mencionado en proyectos anteriores este nos llama la atención a los tres, ya que esta es una herramienta fácil de usar y a los tres nos gusta trabajar con ella, por lo tanto todo el proyecto está programado con ayuda de este Ide.

4. Descripción de la solución

Como se mencionó al inicio del documento escogimos Allegro 5 porque teníamos varias opciones como ncurses, Glade, GTK, QT pero estas se veían más complicadas de usar y de entender además la documentación era más limitada, Allegro 5 tenía una gran cantidad de tutoriales y explicaciones de para qué sirven cada función y al esta ser una biblioteca para desarrollar juegos se podían realizar gran cantidad de funciones del programa con esta, por eso la razón de elegirla.

El proyecto va a contener un Makefile y un archivo de configuración,estos nos permite poderlo compilar y darle las entradas necesarias para que empiece a ejecutar el juego, en el archivo de texto vendrá la velocidad de los fantasmas, la velocidad de PAC-MAN y la disposición de las semillas especiales, que son las que permiten a PAC-MAN cazar a los fantasmas, puede ser normal o aleatorio. Y el último número indica la cantidad de dichas semillas especiales. Como se menciona en las especificaciones del proyecto, existen dos maneras de distribuir las semillas: normal y aleatorio. En el caso de normal, lo que hicimos fue crear un arreglo de 32 números, es decir 16 pares ordenados, donde las posiciones con índices pares son los ejes x e impares para los ejes y. El programa colocará la cantidad de semilla especificada en el tablero. Por otro lado, a la hora de colocar las semillas de forma aleatoria, se necesita generar dos números x, y aleatoriamente, y se verifica si existe la posibilidad de colocarla y de lo contrario se debe generar los números hasta que sea posible de colocarla.

Para mostrar el mapa del juego se creó una matriz de tamaño 20x29 donde contiene diferentes símbolos que representan cada objeto, donde significan:

˜ : Este van a representar las paredes del mapa

“c” y ”||” :

Representan los cocos que come Pac-man, la segunda es una manera de que Inky al ser su movimiento aleatorio puede quedarse estancado en algún lugar, de esta manera con un ese simbolo, le decimos que si encuentra uno de estos genere una nueva dirección pero para toda la demás implementación del juego este símbolo representa un coco.

“S” Significa las semillas que si Pac-man come va a permitir poder matar a los fantasmas. Cada vez que se come una semilla, los fantasmas se cambian de color , y el Pacman tiene 10 movimientos para comerselos.

“ ” El espacio en blanco representa la cueva de los fantasmas donde estos van a estar colocados al inicio del juego.

Como se mencionó antes Pac-Man va a tener 5 vidas, cuando se gana aparece de nuevo el mapa como al inicio del juego pero los fantasmas aumentan su velocidad a un 10 porciento, cuando un fantasma toca a pacman, este pierde una vida y todos regresan a sus posiciones originales. Si pacman come una semilla especial este puede comerse a un fantasma y si se lo come regresa a su cueva a una velocidad de 120 porciento.

Existen 4 fantasmas las características que estos tienen en nuestro programa son:

Blinky es el fantasma rojo, donde es el más inteligente de los cuatro, este va a buscar la ruta más corta para llegar a Pac-man pero va a predecir la ruta de Pinky y trata de encontrar otro camino para poder atrapar al héroe.

Inky es el fantasma más sencillo donde este simplemente se mueve aleatoriamente por el mapa. Para esto se pregunta su dirección actual y si el número random que sería entre 0 y 3 indica hacia dónde se va a mover el fantasma por medio de sus coordenadas x,y. Esto depende también de la dirección que lleva ya que se valida si esa nueva posición es una pared o no.

Nota: Las direcciones en todo el proyecto se trabajaron de la siguiente manera: Para subir la dirección se presenta con un 0, bajar con un 1, izquierda con el número 2, para la derecha un 3, y un 4 significa no se mueva de esta manera evita que se pueda pasar por una pared.

Pinky este por medio del algoritmo implementado busca moverse a la casilla más cercana al Pac-Man, pero tiene una velocidad igual a los demás fantasmas, en cambio Clyde tiene la misma inteligencia que Blinky pero este se mueve un 80 porciento con respecto de los demás fantasmas.

Los fantasmas Blinky y Clyde tienen la misma inteligencia, ellos pueden predecir los movimientos de Pinky y coordinar con ella para poder bloquear a Pac-man en la otra salida del pasillo. Para esto lo que hicimos fue utilizar la ruta más corta entre Pacman y Pinky, y tomamos la posición que se necesita pasar Blinky antes de llegar a Pacman, y el par ordenado de Pacman. Utilizamos esos valores, y lo restamos para ver hacia cuál dirección se va a ir Pinky y así podemos saber por cual lado se deben ir Blinky y Clyde para bloquear a Pacman. Por otro lado, utilizamos un contador para ir calculando cuántas casillas del mismo eje y o eje x se debe pasar el fantasma Pinky para llegar donde se encuentra Pacman. Luego, utilizando ese valor, y le sacamos módulo 5 y le sumamos esto con x o y de Pacman para calcular cuál sería la posición de la casilla que se debe ir los otros fantasmas para bloquearlo. Para esto,también se debe verificar si la casilla es muro o no.

Decidimos utilizar el algoritmo de Floyd-Warshall para encontrar el ca-

mino más corto entre dos nodos. La matriz original de nosotros es de 20 x 29, es decir que tenemos 580 nodos en total, cada nodo está representado con un número de 0-579. Solo 240 nodos de ellos están conectados con al menos un nodo, entonces creamos dos matrices de adyacencia de 240x240, donde en la primera se guardarán las distancias más cortas entre cada par de vértice y la otra almacenará el recorrido. Primero que todo, nosotros decidimos utilizar un struct que contendrá las siguientes informaciones de los 240 nodos en un arreglo: el índice (0-239) , numNodo(0-579),las posiciones x, y en el map, y un arreglo que almacenará cuáles son los nodos que están conectados con cada uno, en este caso máximo solo puede ser 4. En la matriz que se va ir almacenando el camino se pone -1 cuando la distancia entre ellos es infinita. Luego, se inicializa la matriz de distancia utilizando las informaciones que fueron almacenadas en el arreglo mencionado anteriormente. Se pone 0, cuando es la distancia del propio nodo, luego 1 si está relacionado, e infinito en el caso de que no esté conectado.

El algoritmo de Floyd-Warshall consiste en comparar todos los posibles caminos a través del grado entre cada nodo aplicando la técnica de Programación Dinámica, y en ese caso el conjunto de vértices esta numerados de 0 a 239 y existe dos opciones para recorrerlo, el primero sería solo utilizar los vértices del conjunto 1 a k, y luego se verifica desde i hasta k+1, y finalmente de k hasta k, y se verifica si la distancia actual DIST[i][j] es mayor que la distancia DIST[i][k] + DIST[k][j], si es verdad se actualiza en DIST[i][j] . Finalmente, la función de encontrar ruta, que va a recibir dos parámetros el número del nodo inicio y el de nodo destino, y guardará el camino más corto en un arreglo que se va a utilizar cuando los fantasmas se necesitan volver a la cueva cuando se mueren.

Pacman tiene 5 vidas, cada vez que choca contra un fantasma se le restará una vida y todos los personajes del juego se volverán al punto de inicio, y el juego termina cuando el Pac-man ya no le quedan vidas. Para esto decimos no utilizar la ruta más corta, ya que es más sencillo de ver si el fantasma fue comido por Pac-man o fue Fantasma que logró comer al Pac-man. Del mismo modo, el juego original sucede lo mismo. Para simular que los fantasma escapan de pacman lo solucionamos cambiando el destino por una posición fija de la cueva, de esta manera los fantasmas escapan de Pacman pero es poco probable que lleguen a la cueva antes de que se acabe el tiempo para comérselos, cuando este tiempo finaliza su destino vuelve a cambiar para seguir persiguiendo a Pacman.

El juego se reinicia cada vez que el Pacman logra comer todos los cocos

y semillas que se encuentran en el tablero, y la velocidad de los fantasmas se incrementará un 10 porciento. Para esto, se debe llenar el tablero y colocar las semillas y los cocos de nuevo, los personajes se volverán a sus posiciones iniciales, y cantidad de movimientos que quedan para comer fantasmas se iniciará en 0 también.

Para el movimiento de los personajes se utilizaron hilos independientes, con la ayuda de mutex permite que estos lleven velocidades diferentes, donde Blinky, Inky, y Pinky tienen la misma velocidad siempre y su mutex será semf, Pacman también tendrá una velocidad independiente y su mutex será semp, y por último la velocidad de Clyde es un 80 por ciento por lo que multiplicamos la velocidad original por 0.8 y su mutex se representa con semc.

Uno de los mayores problemas que tuvimos fue con el tiempo, ya que era difícil ver la diferencia entre los tiempos de cada personaje, y todos parecían ir a la misma velocidad aunque fueran velocidades independientes. También cuando las semillas se distribuyen aleatoriamente y estas pueden quedar muy juntas, lo que podría dificultar más el juego.

5. Resultados de pruebas

Para este proyecto las pruebas se enfocan en mostrar cómo se desarrolla el juego mientras se va utilizando.

5.1. Pruebas-Cantidad/Tipo de semillas

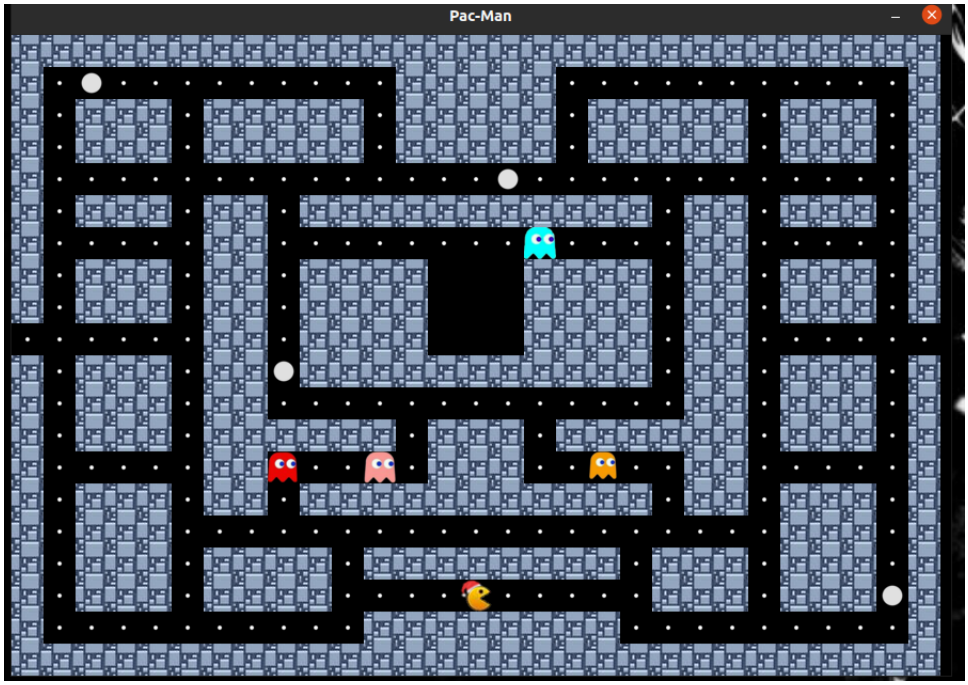

Figura 1: 4 semillas-normal

La primera prueba que se realiza es como entrada recibe una cantidad de 4 semillas y estas se van a distribuir de forma normal. El rango de semillas que se pueden colocar son de 4 a 16, por lo que se puede notar con 4 quedan bastantes separadas unas de las otras, ya que están distribuidas uniformemente. Lo que se puede concluir que entre menos semillas más complicado es el juego ya que cuesta más poder comerse a los fantasmas.

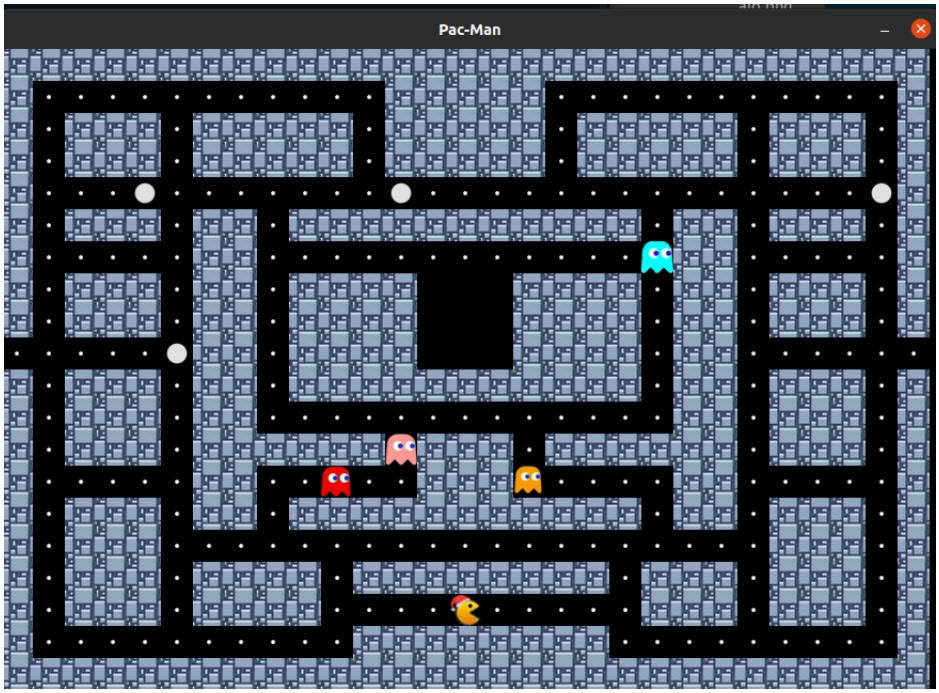

Figura 2: 4 semillas-aleatorio

En esta segunda imagen se vuelven a colocar 4 semillas pero esta vez de manera aleatoria, lo cual se puede notar que al estas ser totalmente random todas pueden quedar en la parte arriba del tablero, mal distribuidas lo que igual dificulta si pacman va por la parte de abajo y si ocupa ayuda no va a encontrar una semilla especial que le permita comer a un fantasma.

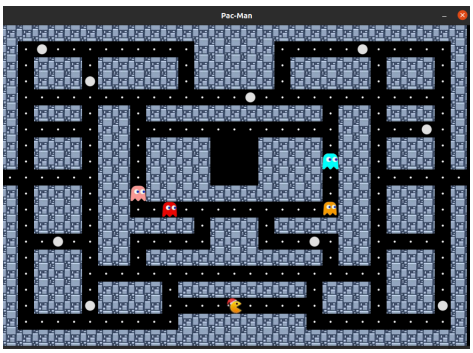

Figura 3: 10 semillas-normal

Ya con 10 semillas distribuidas uniformemente se va a poder jugar por más tiempo ya que permite poder comer más rápido a los fantasmas antes de que se le acabe el tiempo y de esta manera podría permitir pasar al siguiente nivel.

Figura 4: 10 semillas-aleatorio

En esta prueba se busca mostrar la diferencia entre una cantidad pequeña y una grande que se distribuyen aleatoriamente, ya que aquí el problema de que todas queden en un sector cuesta más o se nota menos.

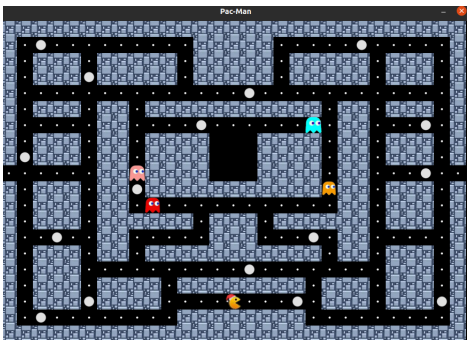

Figura 5: 16 semillas-normal

Ya aquí tiene las 16 semillas, lo cual se busca mostrar cuál sería el máximo que se pueden tener en este tablero, y como estaban distribuidas.

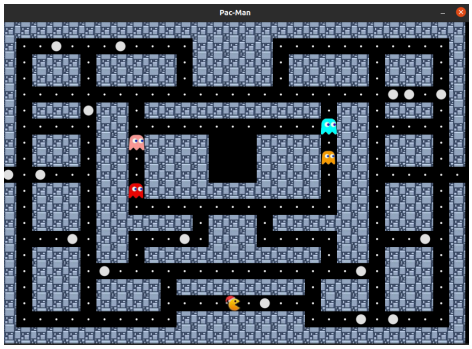

Figura 6: 16 semillas-aleatorio

Esta es igual que la anterior lo que se diferencia es que están distribuidas aleatoriamente, como se puede ver ya en este caso la diferencia entre ambas es mínima por lo que en este caso no influye mucho escoger entre alguna de las dos.

5.2. Pruebas-Velocidades de los fantasmas

Por otro lado, decidimos intentar jugar intercambiando las velocidades de los personajes y la manera de distribuir las semillas en el juego, donde en la figura 7 la velocidad de los fantasmas es de 60 y la de Pac-Man 200, y en la imagen 9 están al revés. A la hora de jugar, notamos una mini diferencia cuando hicimos esos cambios con respecto a la hora de mover al Pac-man ya que al ser más lento nos costaba un toque más de moverlo. Del mismo modo, en las figuras 9-10 como las semillas se distribuyen aleatoriamente fue más difícil de jugarlo.

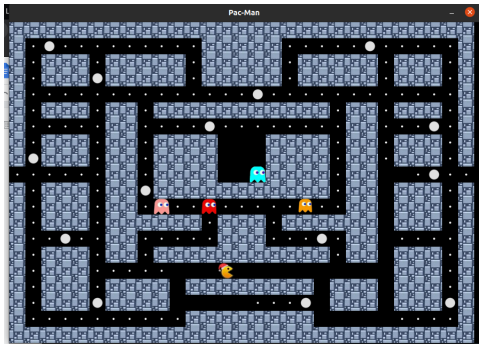

Figura 7: 60-200-normal-15

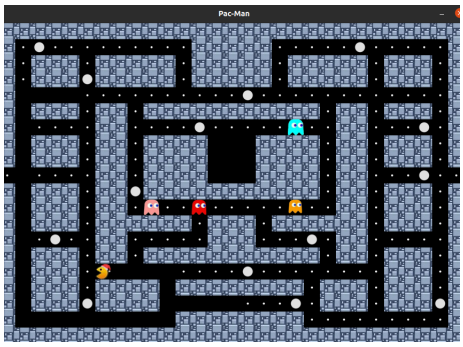

Figura 8: 150-200-normal-15

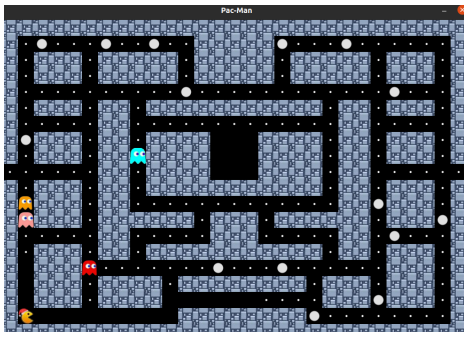

Figura 9: 200-60-normal-15

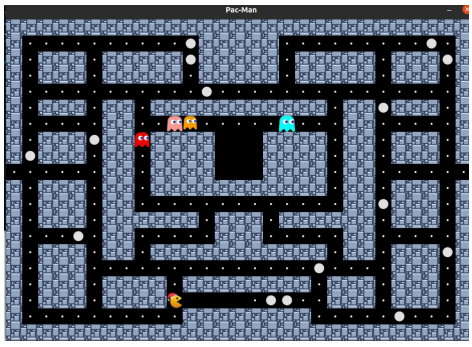

Figura 10: 200-150-normal-15

5.3. Inteligencia Blinky

En las siguientes figuras (11-13), se puede observar que la velocidad de Clyde es más lenta que los otros dos fantasmas y también se puede apreciar que los fantasmas Blinky y Clyde coordinan con Pinky y tratan de bloquear al Pac-Man por otro lado.

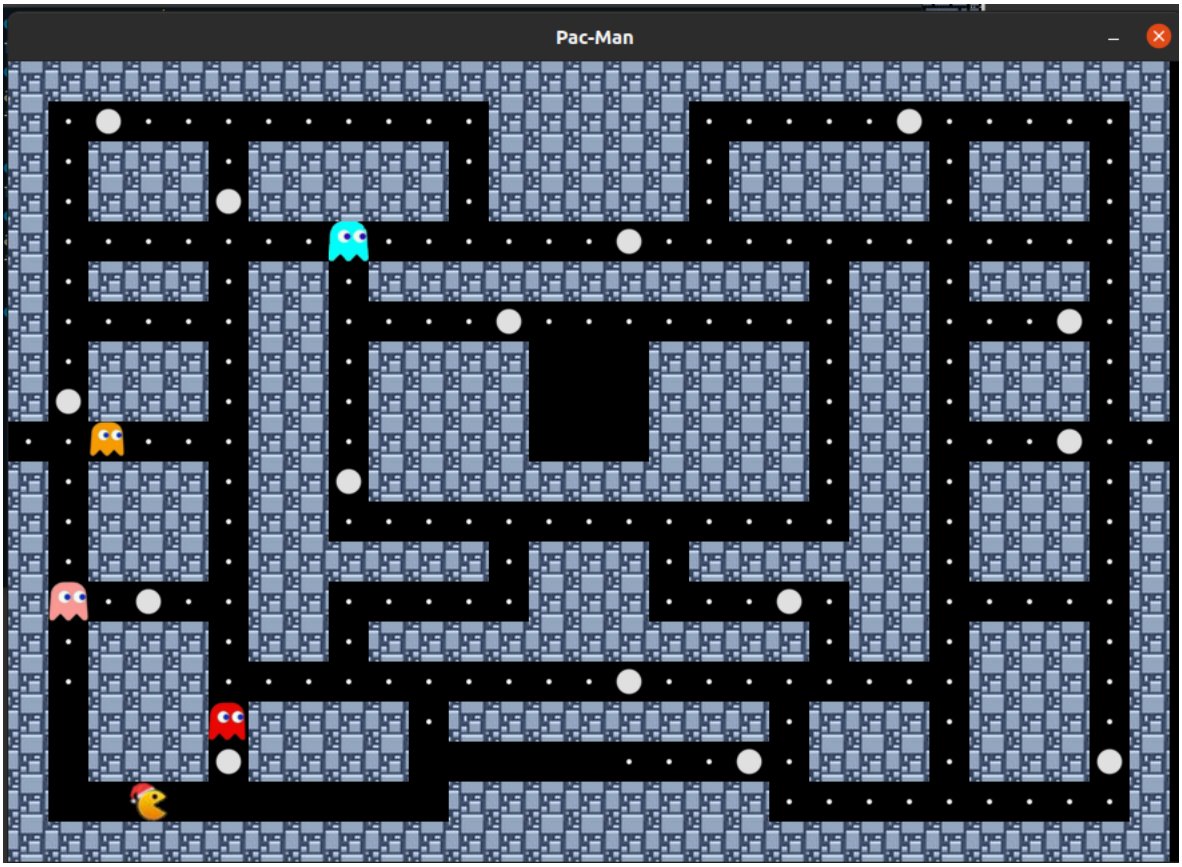

Figura 11: InteligenciaBlinky1

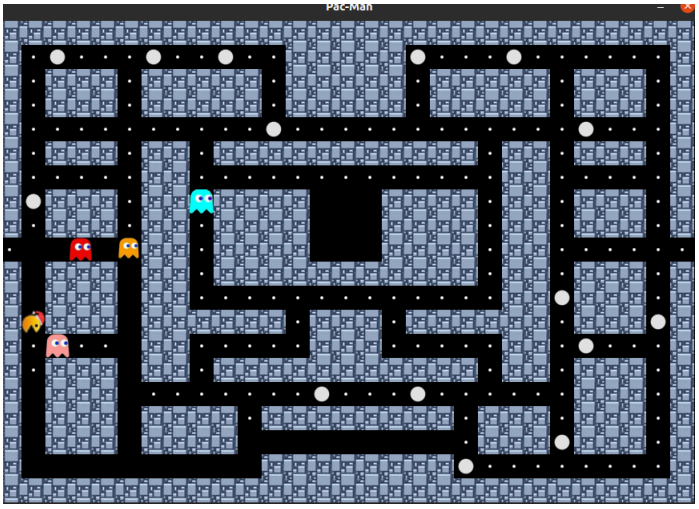

Figura 12: InteligenciaBlinky2

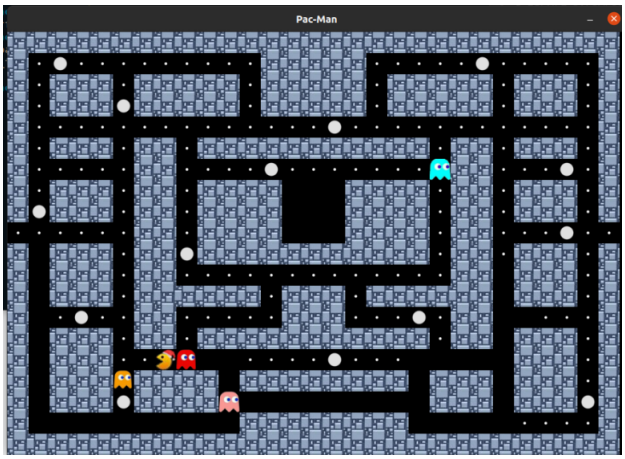

Figura 13: InteligenciaBlinky

5.4. Videos

En ese link [1], se puede ver cuando el Pac-Man come la semilla, los fantasmas se cambian de color y salen huyendo a la cueva. Del mismo modo, al final del video se puede ver como los fantasmas tratan de bloquear al Pac-Man.

Debemos admitir que los 3 somos muy malos jugando Pac-Man, ya que siempre perdíamos muy rápido. Por esa razón decimos grabar el siguiente video [2] donde se puede ver cuando Pac-Man logra comer todas las semillas y que se pasa al siguiente nivel donde las velocidades de los fantasmas se incrementan y que el tablero se reinicie correctamente(Nos costó mucho grabar ese video).

[1] Puede verse aquí
[2] Puede verse aquí

6. Conclusiones

Para concluir, debemos admitir que fue un proyecto largo, donde pasamos de ver pocas líneas de código en Scheme, Erlang y Prolog a ver una gran cantidad en C. Fue un proyecto divertido y bonito de programar donde se pudieron aplicar diferentes conocimientos en el Lenguaje de programación C.

Hay detalles que no logramos cubrir, pero estamos orgullosos del trabajo realizado, donde como en todo proyecto tuvimos problemas pero tratamos de resolverlos de la mejor manera. Por medio de las pruebas se pudo observar la dificultad que puede tener el juego según las distribución y cantidad de semillas que se pongan, al igual influye la velocidad de los personajes, la cantidad de tiempo para comerse a los fantasmas y la cantidad de vidas que se pueden tener.

De las pruebas realizadas se puede observar los diferentes comportamientos que tiene a lo largo del juego, donde cada detalle es importante y se puede apreciar. Esperamos que con este programa se pueda abarcar los objetivos que tenía este curso con respecto a este lenguaje y lo más importante fue que como grupo aprendimos bastante y nos divertimos.

7. Aprendizajes

7.1. María José Barquero Pérez

Este proyecto fue interesante ya que fue una mezcla de todo porque fue bonito programar un juego pero al mismo tiempo este lleva muchos detalles y se debe tener mucho cuidado. Puede reforzar mis conocimientos en C, aprendí que existe una biblioteca llamada allegro 5 que facilita programar juegos, como realizar una interfaz en C y lo más importante fue que nos divertimos programando, excepto cuando teníamos problemas con algunos rangos y tiempos.

7.2. Te Chen Huang

En este proyecto, pude aplicar la mayoría de los conocimientos aprendidos en este curso, aprovechando al máximo las ventajas que tienen el lenguaje C. Del mismo modo, conocí y aprendí a utilizar la librería Allegro 5 para la parte de interfaz. Logré entender mejor y reforzar la teoría del algoritmo Floyd Warshall, Programación Dinámica y el uso de hilos. Igual que los otros proyectos que se han realizado a lo largo del curso, es importante entender el problema y discutir las posibles soluciones en grupo. Debo admitir que ha sido uno de los proyectos más lindos y divertidos que he tenido en estos dos años.

7.3. Kendall Rodríguez Mora

Este proyecto fue muy divertido e interesante, reforcé mis conocimientos en cuanto a programación estructurada, nunca había programado en C un proyecto, me gusto bastante C, al menos lo que pude llegar a usar. Tampoco había utilizado la biblioteca Allegro, ni ninguna biblioteca diseñada específicamente para videojuegos, no vi que fuese muy complicado utilizar la biblioteca por su útil documentación en internet. Este proyecto en lo personal me gusto mucho ya que uno lo pasa mejor programando algo que le gusta o que le divierte, por lo que no se sentía como si fuese un proyecto aburrido o tedioso.

Referencias

[1] Allegro 5. *Allegro 5 reference manual*. Web; Accedido el 02-12-2020. 2019. URL: https://liballeg.org/a5docs/trunk/.

[2] Los diccionarios y las enciclopedias sobre el Académico. *Stdlib.h*. Web; Accedido el 12-12-2020. URL: https://www.citethisforme.com/es/cite/sources/websiteautociteeval.

[3] Allegro. *Graphics routines*. Web; Accedido el 02-12-2020. URL: https://liballeg.org/a5docs/trunk/graphics.html#al_map_rgb.

[4] Anónimo. *Algoritmo de Floyd-Warshall*. Web; Accedido el 05-12-2020. URL: https://riptutorial.com/es/dynamic-programming/example/25781/algoritmo-de-floyd-warshall.

[5] Anónimo. *Passing arguments to pthread function*. Web; Accedido el 04-12-2020. URL: http://www.cse.cuhk.edu.hk/~ericlo/teaching/os/lab/9-PThread/Pass.html.

[6] Anónimo. *Programación de Juegos y Gráficos en C con Allegro*. Web; Accedido el 02-12-2020. URL: https://personales.unican.es/corcuerp/progcomp/slides/Allegro.pdf.

[7] CodingMadeEasy. *C++ Allegro 5 Made Easy Tutorial 1 - Intro Install*. Web; Accedido el 02-12-2020. 2012. URL: https://youtu.be/IZ2krJ8Ls2A.

[8] GeeksForGeeks. *Finding shortest path between any two nodes using Floyd Warshall Algorithm*. Web; Accedido el 05-12-2020. 2020. URL: https://www.geeksforgeeks.org/finding-shortest-path-between-any-two-nodes-using-floyd-warshall-algorithm/.

[9] GeeksForGeeks. *Floyd Warshall Algorithm — DP-16*. Web; Accedido el 05-12-2020. 2019. URL: https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/.

[10] Kishori Mundargi. *Historia de la programación en C*. Web; Accedido el 11-12-2020. 2003. URL: https://www.peoi.org/Courses/Coursessp/cprog/cprog1.html.

[11] Tom Riecken. *Learn About C Programming: Why This Language Still Rules*. Web; Accedido el 11-12-2020. 2020. URL: https://www.whoishostingthis.com/resources/c-developer/.

[12] Universidad en Segovia. *Hilos de ejecución de POSIX*. Web; Accedido el 05-12-2020. 2004. URL: https://www.infor.uva.es/~fdiaz/so/2004_05/doc/SO_PR05_20041026.pdf.