PACe-MAN

Greivin Carrillo Rodríguez

Fabián Castillo Cerdas

Jorge Guillén Campos

Instituto Tecnológico de Costa Rica

Profesor Marco Rivera Meneses

Paradigmas de programación

10 de noviembre de 2023

Índice

Descripción del uso de las estructuras de datos desarrolladas

En la implementación de la tarea se definieron estructuras como listas y grafos para almacenar los datos de posiciones de dots y posibles espacios para la aparición de objetos

como frutas o pastillas; y guarda la relación entre nodos para validar posibles movimientos: respectivamente:

Lista: se crea una lista para representar posiciones en el mapa, esta lista va desde 1 hasta 162. Al inicio cada casilla se le coloca un dot, posteriormente si el administrador quiere colocar un objeto como una pastilla o una fruta deberá validar la posición en la que se desea colocar dicho objeto, luego, en el llamado a la función que crea el objeto, uno de los parámetros requeridos es esa posición.

Grafo: En nuestra implementación se utilizan el grafo como concepto ya donde realmente se almacena la información es en una lista, pero dicha lista se puede representar con un grafo no dirigido. Este grafo nos funcionara para posteriormente encontrar la ruta mas corta mediante el algoritmo de Dijkstra, para el movimiento de los fantasmas. La lista esta conformada por 64 nodos los cuales están colocados en ramificaciones del mapa donde se crean posibles caminos, de los cuales se escoger la ruta más corta para que el fantasma cumpla con su objetivo, cabe recalcar que los fantasmas tienen distintos movimientos, pero la base del algoritmo de búsqueda es la misma.

Nodos: se utilizan nodos para guardar información específica de un nodo perteneciente al grafo como la posición X y Y, el nombre y los nodos adyacentes.

Plan de Trabajo

Actividad	Encargado			Fecha
	Greivin	Fabian	Jorge	
Dividir el proyecto	x	x	x	21-oct
Definir estructuras de datos	x	x	x	
Investigacion sobre servidor		x		24-oct
Investigacion sobre cliente	x		x	
Creacion de Servidor y sus metodos		x		27-oct
Creacion de Cliente (Jugador)	x			28-oct
Conexion Cliente-Servidor		x	x	1-nov
Creacion de Cliente (Observador)			x	4-nov
Creacion de control fisico			x	7-nov
Correccion de errores	x	x	x	8-nov
Implementar solucion final	x	x	x	
Hacer manual de usuario		x		9-nov
Hacer anexo			x	
Hacer la documentacion	x	x		
Revision final	x	x	x	10-nov

Conclusiones

El desarrollo de PACe-Man se ha llevado a cabo con el objetivo de fortalecer el conocimiento de los paradigmas de programación imperativo y orientado a objetos. A través de la implementación de una aplicación que combina la lógica en C y Java, este proyecto ha logrado alcanzar sus objetivos de manera efectiva.

La aplicación servidor, programada en C, actúa como un administrador del juego, aplicando los principios de programación imperativa para gestionar la lógica del juego. Por otro lado, la aplicación cliente en Java utiliza los conceptos de programación orientada a objetos para proporcionar una interfaz gráfica interactiva y dinámica para los jugadores.

La creación y manipulación de elementos en el juego, como frutas y fantasmas, ha demostrado la aplicación práctica de los conceptos de programación imperativa. La asignación de vidas, el control de niveles y la gestión de la puntuación son ejemplos claros de cómo estos paradigmas se han integrado de manera efectiva para ofrecer una experiencia de juego completa.

Además, la aplicación cliente ha implementado la capacidad de gestionar múltiples jugadores, observadores y la comunicación eficiente entre el servidor y los clientes, destacando la versatilidad de las estructuras de datos utilizadas en el proyecto.

Recomendaciones

Durante el desarrollo de PACe-Man, se han identificado recomendaciones y áreas para futuras actualizaciones que pueden mejorar la aplicación y la experiencia de juego:

1. Interfaz Gráfica de Usuario (GUI) Mejorada: Actualmente, PACe-Man utiliza una interfaz gráfica en Java, pero se podría considerar una mejora adicional para hacerla más visualmente atractiva y amigable. Incorporar elementos gráficos y una presentación más intuitiva podría mejorar significativamente la experiencia del jugador.
2. Expansión de Características: Considerar la adición de nuevas características y elementos de juego para mantener la experiencia fresca y emocionante. La introducción de desafíos adicionales, power-ups o modos de juego podría agregar variedad y aumentar la jugabilidad.
3. Optimización del Rendimiento: A medida que el juego evoluciona, es crucial optimizar el rendimiento para garantizar una experiencia de juego fluida. La implementación de técnicas avanzadas, como la gestión eficiente de recursos y la optimización de algoritmos, puede contribuir a un rendimiento óptimo.
4. Validación Mejorada de Entradas del Usuario: Reforzar la validación de entradas del usuario para manejar una variedad más amplia de comandos y solicitudes. Esto asegurará que la aplicación sea robusta y pueda gestionar diferentes interacciones de los jugadores de manera efectiva.
5. Documentación Detallada: Mejorar la documentación del proyecto, proporcionando manuales de usuario claros y técnicos detallados. Esto facilitará la comprensión y el desarrollo continuo del proyecto, permitiendo a nuevos desarrolladores y usuarios aprovechar al máximo PACe-Man.
6. Pruebas Extensivas: Realizar pruebas extensivas en diversas plataformas y situaciones de juego para garantizar la estabilidad y confiabilidad del juego. Las pruebas deben abordar escenarios de juego variados y asegurar que todas las funciones respondan como se espera.
7. Colaboración y Contribuciones Comunitarias: Considerar abrir el proyecto a la colaboración de la comunidad de desarrolladores. Invitar a otros a contribuir con ideas, correcciones de errores y nuevas características puede enriquecer la aplicación y fomentar un desarrollo colaborativo.

Bibliografía

1. Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language (2nd ed.). Prentice Hall.

2. King, C. S. (2008). C Programming Absolute Beginner's Guide (3rd ed.). Que Publishing.
3. Horstmann, C. S., & Cornell, G. (2013). Core Java Volume I--Fundamentals (9th ed.). Prentice Hall.
4. Sierra, K., & Bates, B. (2018). OCA/OCP Java SE 8 Programmer Certification Kit: Exam 1Z0-808 and Exam 1Z0-809. Sybex.
5. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
6. Martin, R. C. (2003). Agile Software Development, Principles, Patterns, and Practices. Prentice Hall.
7. Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). UNIX Network Programming, Volume 1: The Sockets Networking API (3rd ed.). Addison-Wesley.
8. Beej's Guide to Network Programming (Online Resource): http://beej.us/guide/bgnet/

Bitácora en digital

Greivin:

- Se hizo una reunión para dividir las partes del trabajo y para declarar como ir trabajando. (21 octubre)
- Cree un plan de actividades a realizar para el proyecto. (23 octubre)
- Investigue sobre el lenguaje Java para la parte del cliente. (24 octubre)
- Cree la clase jugador con sus funcionalidades de movimiento básicas. (28 octubre)
- Trabaje en las colisiones del jugador con el mapa. (29 octubre)
- Trabaje en los movimientos de los fantasmas para su implementación en el servidor. (2 noviembre)
- Realice los distintos movimientos para el jugador cuando obtiene una pastilla y para los fantasmas al huir del jugador. (4 noviembre)
- Trabaje en la documentación. (9 noviembre)
- Se hizo una reunión para finalizar detalles y revisar el código final. (10 noviembre)

Fabian:

- Se realizo una reunión inicial para definir la metodología del trabajo. (21 octubre)
- Investigue sobre implementaciones en C con juegos similares. (26 octubre)
- Busque los sprites para los objetos del juego junto posibles matrices para el mapa. (27 octubre)
- Cree una lista con la información del mapa, posiciones de los dots y futuros objetos a implementar. (27 octubre)
- Cree los métodos para la creación de fantasmas, manejo de vidas, creación de frutas y pastillas. (29 octubre)
- Cree los métodos para la velocidad de los fantasmas y para la puntuación del jugador. (31 octubre)
- Implemente el movimiento de los fantasmas (3 noviembre)
- Solucione bugs con respecto al cliente observador (6 noviembre)
- Realice la sección de recomendaciones y conclusiones para la documentación. (9 noviembre)
- Realice el manual de usuario. (9 noviembre)

- Se realizo una reunión para finalizar el código y concluir con la solución del proyecto. (10 noviembre

Jorge:

- Nos reunimos para dividir el trabajo y como trabajarlo. (21 octubre)
- Investigue sobre interfaces en Java y comunicaciones con el lenguaje C. (24 octubre)
- Trabaje en la actualización de la posición del jugador en la estructura de datos utilizada. (27 octubre)
- Modifique la clase jugador para optimizar el reconocimiento de colisiones con el mapa. (30 octubre)
- Trabaje en la conexión cliente servidor en el sistema operativo Linux (1 noviembre)
- Cree un cliente observador que toma los datos de un jugador ya existente. (4 noviembre)
- Cree un control físico para el manejo del Jugador. (7 noviembre)
- Trabaje en la conexión del control con el jugador para su funcionalidad (8 noviembre)
- Nos reunimos para revisar el código. (10 noviembre)