Instituto Tecnológico de Costa Rica

Area académica de Ingeniería en Computadores

II Semestre 2019

Tarea 3. Paradigma imperativo y orientado a objetos.

Nombre código: DonCEyKongJr

Equipo de trabajo

- **Jesús Sandoval Morales** [2018203706] - *Desarrollador* - shuzz22260427
- **Alejandro Ibarra Campos** [2018089951] - *Desarrollador* - AlejandroIbarraC
- **Esteban Alvarado Vargas** [2018109336] - *Desarrollador* - estalvgs1999

Responsable Técnico:

- **Marco Rivera Meneses** - *Profesor del Curso*

Curso:

Lenguajes, Compiladores e Intérpretes - CE 3104

Lenguajes utilizados:

C y Java

Nota: El código también se puede encontrar en el repositorio de GitHub https://github.com/AlejandroIbarraC/DonCEyKongJR

E

A. Descripción del problema

DonCEyKongJr corresponde a la tercera tarea programada del módulo de Lenguajes, correspondiente al curso de Ingeniería en Computadores “Lenguajes, Compiladores e Intérpretes - CE 3104”.

Se debe programar un juego, un clon de DonkeyKongJr, el clásico juego de plataformas de Nintendo, para ligar los lenguajes de programación C y Java, para finalizar el estudio de los paradigmas de programación imperativo y orientado a objetos.

El juego debe tener las principales características:

- El servidor mantiene un control remoto de los enemigos que aparecen en el juego. Crea y elimina frutas, y es capaz de lanzarle dos tipos de cocodrilos al jugador: rojos y azules, con comportamientos específicos que se describirán más adelante. El mismo es implementado por sockets desde Java.
- El cliente está implementado en C, Donkey Kong Jr se mueve con las teclas direccionales y salta de liana en liana hasta llegar a la parte superior de la pantalla. Si toca un lagarto, muere. Además, existe la opción de espectar una partida desde otro cliente, sin interferir con el control de la que está en curso.

B. Objetivos

Objetivo General:

- Reafirmar los conocimientos obtenidos en clases acerca del paradigma imperativo y orientado a objetos por medio del desarrollo de un juego que involucre el ligue en software entre dos tipos de lenguajes que cumplan estas características.

Objetivos específicos:

- Plantear una solución implementada en C y Java; un juego basado en el juego de plataformas DonkeyKongJr de Nintendo.
- Aplicar los conceptos de programación imperativa y orientada a objetos en la resolución del problema general, con sus características particulares.
- Desarrollar un sistema cliente-servidor para la comunicación sencilla y el envío de información, que permita una fácil y eficiente interacción entre las diferentes partes del programa.

C. Estructuras de datos desarrolladas y utilizadas.

Para el desarrollo del susodicho programa se utilizaron una serie de estructuras de datos, las cuales tuvieron como finalidad facilitar el manejo tanto lógico como gráfico del juego.

- **Listas Enlazadas**:

Las listas enlazadas fueron desarrolladas en la parte del código designado a C, esto debido a que en dicho lenguaje se implementó la lógica del juego como tal.

El uso principal que tuvieron estas estructuras fue el de facilitar el manejo de los enemigos y la aparición de frutas, desde su posicionamiento hasta el movimiento, en el caso de los cocodrilos.

El concepto general de una lista enlazada se ilustra en la siguiente figura, el mismo sigue el estándar basado en otros lenguajes. En el caso de C, cada nodo es una estructura propia “struct”, y las relaciones punteros a diferentes posiciones de memoria.

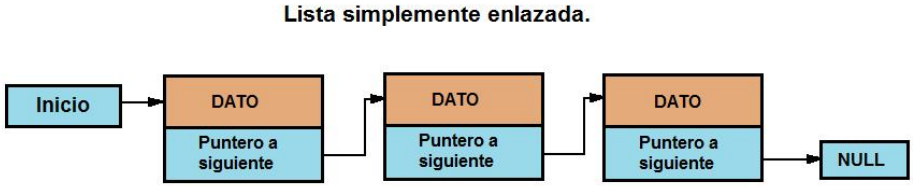


Figura 1. *Esquema de una lista enlazada.*

El modo en el que se aplicó a la lógica se divide en listas para enemigos y listas para frutas. A continuación, se procederá a explicar cada caso:

1. **Enemigos:**

En el caso de los enemigos, es necesario manejar una lista enlazada que los pudiese almacenar de forma dinámica. Para esto, en cada nodo se almacena un cocodrilo enemigo, y se programaron los métodos de eliminar y agregar un enemigo de la lista por medio de nodos.

En el flujo del programa, cada vez que se actualiza la matriz, la lista es recorrida por un ciclo while, el cual revisa una serie de casos a tomar en cuenta. Por ejemplo, si el enemigo está sobre una plataforma, se moverá en una dirección hasta tocar una pared o chocar con el jugador. Por medio de estos casos programados con condicionales, se realiza el movimiento y el funcionamiento de cada uno de los enemigos cada vez que se recorre el ciclo.

Se realiza tomando en cuenta cada posible escenario lógico del movimiento, tanto dentro vertical como horizontalmente, con esto se actualizan las coordenadas de la matriz que controlan las apariciones y desplazamientos de los diferentes enemigos.

Ambos tipos de lagarto tienen comportamientos diferentes, el tipo de lagarto rojo se desplaza en vertical por las lianas y se devuelve (arriba-abajo, abajo-arriba), mientras el azul cae al vacío cuando llega al final de la liana (abajo).

2. **Frutas.**

En el caso de las frutas, la lista fue usada por motivos similares a los enemigos, solo que en este caso se controlan las posiciones adyacentes a cada fruta. Esto con el motivo de saber cuando el jugador puede estar próximo a tocarla, en este caso se verifican las posiciones y se desaparece la fruta por medio de un indicador.

- **Matriz**:

El mapa del juego está implementado a manera de matriz. Se escogieron las dimensiones de 24 por 16 cuadros, ya que el original del juego de Nintendo se ajusta correctamente a estas dimensiones. Se adjunta una copia de la matriz que es cargada dinámicamente por el programa en un archivo “matrix.csv”, para su mejor visualización.

Se escogió este método de solución para facilitar el envío de información por sockets, ya que otras soluciones como movimiento individual de pixeles involucran dimensiones de complejidad altas al serializar muchísima información en tiempo real. Se sacrifica detalle visual, pero se hizo para cumplir con la especificación.

En este documento, cada número representa la posición de un elemento diferente.

0 = espacio vacío
13 = liana
11 = suelo
12 = plataforma “scaffolding”
7 = llave para pasar de nivel
411 = jugador, Donkey Kong Jr.
22 = lagarto azul
21 = lagarto rojo

Cada uno de estos elementos, posee diferentes sprites gráficos que representan su estado actual, para lograr una animación fluida de los mismos. La matriz representa en cada instante de la ejecución, el estado actual, y se actualiza lógicamente para luego ser representada gráficamente.

D. Descripción de los algoritmos utilizados.

Para el correcto desarrollo del programa fueron necesarios diversos algoritmos y funciones que cumplieran los roles deseados en cada parte del código. En este apartado se procederá a describir de forma detallada los algoritmos que fueron usados en la creación del juego.

- **Algoritmos de detección de colisiones.**

La función de este algoritmo es el de revisar las coordenadas en i y j en la matriz y verificar si coinciden o superponen con la de una fruta o un enemigo y con base en esto realizar una acción u otra. En el caso de los enemigos matar al jugador, y en el caso de las frutas que se puedan comer

- **Algoritmos de movimientos.**

Este algoritmo funciona por medio del mismo ciclo que el algoritmo anterior, se revisan continuamente las posiciones del jugador y de los enemigos según casos predefinidos por medio de un int y dependiendo de cada caso se realiza un movimiento u otro en el caso de los cocodrilos o se le permite hacer un movimiento o no en caso del jugador.

Ejemplo: Si el jugador choca con un borde, no se puede mover. Si el jugador salta hacia la derecha, se mueve hacia arriba para realizar la animación

- **Algoritmo de creación.**

Este algoritmo se usa para colocar tanto a las frutas como a los cocodrilos en tiempo real o antes de que empiece el juego. Por medio de casos realiza verificaciones de la posición en la cual se desea colocar la fruta y en cocodrilo y

dependiendo del lugar que marquen las posiciones de i y j se trata de una manera o de otra. Recordemos que la matriz es lógica interna.

E. Problemas conocidos.

Durante el desarrollo del programa se lograron identificar una serie de problemas persistentes a la hora de tratar de implementar ciertas acciones lógicas o graficas. Entre dichos problemas cabe destacar los siguientes:

- **Problema con los observadores.**

Estos problemas se dieron principalmente debido a razones de tiempo, pero en el desarrollo final fue sumamente problemático implementar el sistema de conexiones para que múltiples usuarios pudiesen ver el juego.

- **Problemas con movimiento (Visual).**

Este como tal no es un problema, ya que no causa ningún tipo de error; sin embargo, es un daño menor de forma visual. En sí, el error es que a la hora de moverse, debido a la definición matricial, la ejecución tiende a parecer que los elementos se desplazan mediante teletransportación entre los cuadros de la matriz.

- **Desaparición de jugador**

En algunos casos, el jugador desaparece de la pantalla, debido a un error no identificado en la actualización de la matriz. Este problema no fue solucionado en el tiempo pedido.

- **Rendimiento en computadoras con bajas especificaciones**

En computadoras con bajas especificaciones, el código no corre de manera óptima, debido a la cantidad de actualizaciones que suceden en la pantalla. Se supone que este problema sucede debido a la biblioteca de interfaz Allegro.

F. División de tareas para cada miembro del equipo.

Tarea	Jesus	Alejandro	Esteban
Realizar la lógica de colisiones para enemigos y jugador			
Realizar la interfaz en Allegro			
Montar los sockets			
Realizar la interfaz en java			
Parser entre C y Java			
Matriz de movimiento para enemigos y jugador			
Creación y manejo de la lista dinámica para almacenar frutas y enemigos			

Negro = Responsabilidad total.

Naranja = Responsabilidad compartida.

Amarilla = Responsabilidad grupal.

E

G. Problemas encontrados (solucionados).

- **Problemas con los sockets.**

A la hora de realizar el traspaso de datos por medio de sockets ocurrieron una serie de problemas de diferentes naturalezas; no se enviaban bien, se enviaban los datos cortados o de plano no se enviaban. Gratamente después de realizar una exhaustiva investigación en páginas las cuales serán citadas en la bibliografía se pudieron solucionar de forma exitosa, utilizando librerías default de Windows, wisock32.

- **Problemas de almacenamiento en lista.**

A principio cuando se definió la lista con la cual se iban a guardar las frutas y los enemigos se empezaron a generar problemas a la hora de definir los punteros; sin embargo, todos los problemas finales fueron resueltos con éxito después de analizar el código y consular por soluciones alternas.

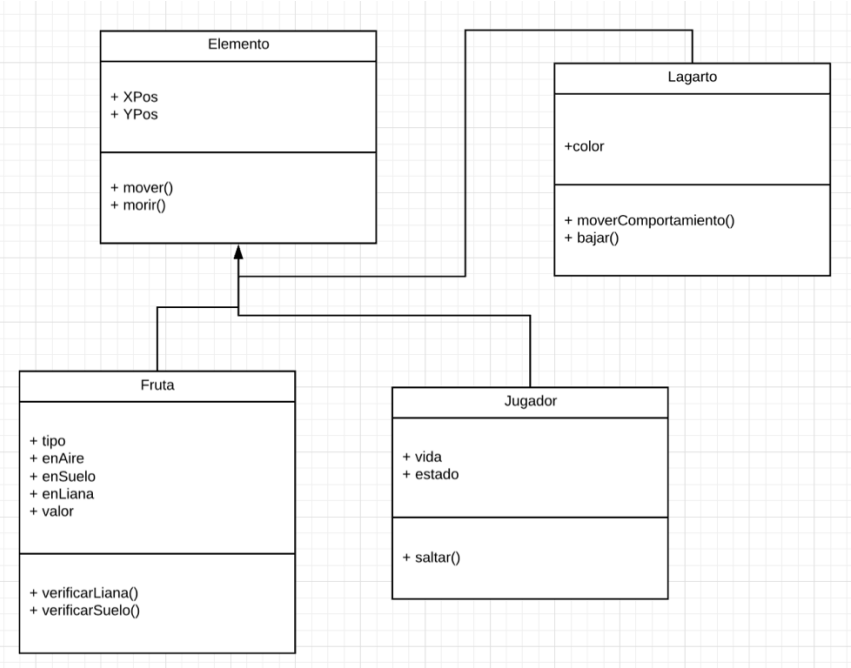
- **Problemas de sistema operativo.**

El desarrollo inicial se dio en el sistema operativo Linux, pero debido a la falta de información para el desarrollo de interfaces en Allegro, se decidió trasladar el sistema operativo de todo el equipo a Windows. Se encontró mayor facilidad el trabajar en C en este sistema.

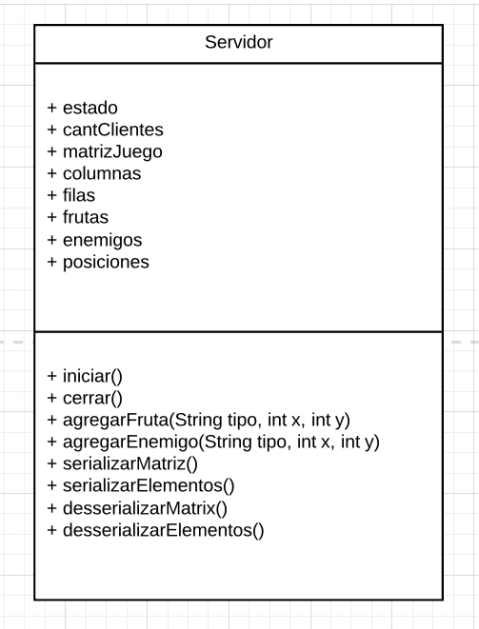
H. Diagramas de clases. (Objetos)

- Inicial

Elementos generales:



Servidor:

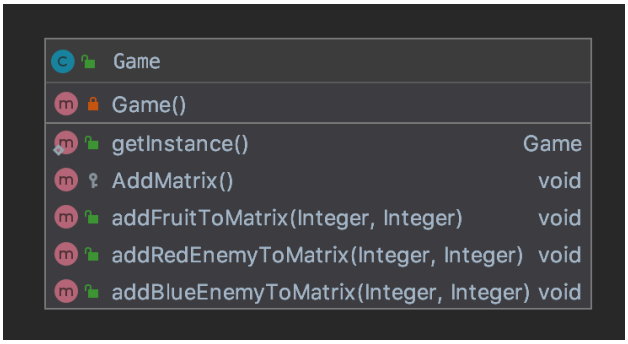


E

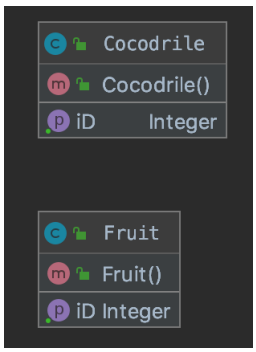
- Final

Ventanas de Interfaz (JavaFX)

Juego final

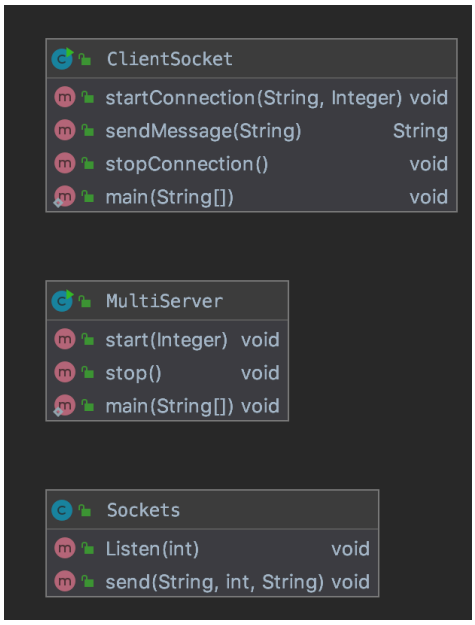


Elementos



E

Sockets y servidor



I. Conclusiones y Recomendaciones.

- Conclusiones.

-En el paradigma imperativo, se debe considerar el uso de memoria.

-Los punteros son herramientas extremadamente útiles para conseguir manejar memoria de forma segura y lógica.

-Es de suma importancia tener clara la teoría al momento de manejar la memoria, esto debido a la cantidad de problemas que esto podría conllevar si se hace de mala manera.

-En C es esencial entender como trabajan las “structs” o estructuras, que agrupan datos asociados a un nombre.

-La programación orientada a objetos es un paradigma sumamente natural por lo cual lo hace muy intuitivo para programar.

- Recomendaciones.

-Realizar programas de la forma mas modular posible, ya que esto facilita tanto la lectura como la implementación del código.

-Sacarles el máximo partido posible a los punteros y aprovechar el poder que estos otorgan para manejar memoria.

-Evitar en la medida de lo posible acciones que puedan corromper la memoria de forma tanto directa como indirecta.

J. Bitácora

17-18 De septiembre de 2019.

Debido a una serie de trabajos, inconvenientes externos por parte de todos los miembros del grupo y exámenes, el proyecto tuvo que ser iniciado hasta el día de hoy.

Tomando en cuenta la falta de tiempo el trabajo se dividió de forma apresurada, pero concisa, se trato de que la división fuera lo mas equitativa posible. En esta división lo primero que se discutió fue la ubicación que se le iba a dar a la lógica, es decir, si se haría en el servidor o en el cliente, a lo cual se llegó a la decisión de hacerlo en el cliente, debido a que esto facilitaba la implementación desde el punto de vista como grupo.

La división de tareas quedo de la siguiente manera:

Esteban: Será el encargado de la parte de conexiones por medio de Sockets, además ayudará a Jesus con la lógica del juego en C

Jesus: Su parte será toda la lógica interna del juego junto a Esteban, así como la del manejo de la matriz y la creación de interfaz en Java bajo la supervisión meticulosa y estética de Alejandro.

Alejandro: Su tarea será la de realizar la interfaz general del juego, la cual será programada en C, además de esto tendrá la tarea de supervisar a Jesus en la interfaz secundaria.

Además de esta división de tareas el mismo día de hoy se empezó a codificar de manera feroz, reuniéndonos en el apartamento de Jesús, ubicado 50 metros sureste del bar la nave desde las 12:21 medio día hasta las 2:09 a.m. del día 18 de septiembre del 2019.

Los avances de hoy fueron los siguientes:

Jesus: Creó la interfaz en java con el fin de agregar objetos, creó la matriz inicial sobre la cual se va a manejar la lógica

Esteban: Realizó la conexión entre los sockets de forma completa

Alejandro: Realizó grandes avances en la interfaz de C, la cual está siendo construida en Allegro

18-19 De septiembre de 2019.

El día de hoy como grupo amanecimos en el aparta de Jesus (su susodicho redactor) a las 7 a.m., cada miembro del grupo procedió a ir a sus respectivas lecciones del día. Posteriormente el grupo se volvió a reunir a las 5:14 p.m. al haber todos salidos de clases y se procedió a iniciar a codificar de nuevo.

E

El grupo se reunió desde las 5:14 p.m. hasta las 3:11 a.m. durante este proceso los avances fueron los siguientes:

Jesus: Empezó el manejo de los movimientos de los enemigos por medio de una matriz que se recorrer en tiempo real, inicializo dicha matriz, creo las listas dinámicas para el manejo de enemigos

Esteban: Logró serializar los datos en forma de matriz para así poder actualizarlo los elementos del juego en tiempo real.

Alejandro: Siguió trabajando en la interfaz, además empezó a crear sprites y diversos arreglos estéticos para el juego.

19-20 De septiembre de 2019.

El día de hoy fue la ultima tanda de tiempo que se tuvo como grupo, debido a esto se decidió aprovechar lo máximo posible y como grupo nos reunimos desde las 12 p.m., mas exactamente a las 12:34 p.m. con el fin de avanzar o, en su defecto avanzar lo máximo posible.

Los avances de hoy fueron los siguientes:

Jesus: Terminó la lógica de movimientos para todos los elementos dentro del juego, así como toda la lógica de posiciones y colisiones

Esteban: Ayudó a Jesus con la lógica de movimiento, terminó el parser de C a Java

Alejandro: Terminó la interfaz con todos los sprites necesarios para animar cada movimiento de forma correcta.

22 de septiembre de 2019.

El último día para la entrega, fue un fin de semana, por lo que se asignó a cada uno sus últimas tareas para el envío final.

Jesus: Agregó métodos finales al servidor en Java, terminó la documentación técnica.

Esteban: Finalizó la conexión entre Java y C, el cliente y servidor, el funcionamiento correcto de cada una de las partes. Trabajó en la serialización y deserialización correspondiente para las actualizaciones de la matriz.

Alejandro: Terminó la organización de los archivos del proyecto, realizó la descripción del manual de usuario, ayudó a Jesús con la documentación técnica, y se le asignó enviar el proyecto al profesor.

I. Bibliografia.

1. Tutorialspoint.com. (2019). Linked List Program in C - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/data_structures_algorithms/linked_list_program_in_c.htm [Accessed 20 Sep. 2019].

2. Jetbrains.com. (2019). Creating diagrams - Help | CLion. [online] Available at: https://www.jetbrains.com/help/clion/creating-diagrams.html [Accessed 20 Sep. 2019].

3. C, b. and Zifkin, B. (2019). building a very simple parser in C. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/18948914/building-a-very-simple-parser-in-c [Accessed 20 Sep. 2019].

4. C, b. and Zifkin, B. (2019). building a very simple parser in C. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/18948914/building-a-very-simple-parser-in-c [Accessed 20 Sep. 2019].