**Instituto Tecnológico de Costa Rica**

**Área de Ingeniería en Computadores**

CE-3104: Lenguajes, Compiladores e Intérpretes

**Profesor:**

Marco Rivera Meneses

**Tarea 3: DonCEy Kong Jr**

Paradigma imperativo y orientado a objetos

**Integrantes:**

Carlos Adrian Araya Ramirez

Michael Shakime Richards Sparks

José Andrés Solano Mora

**Abril, 2021**

**Tabla de Contenidos**

[1. Breve descripción del proyecto 3](#_Toc67018133)

[1.1. Descripción detallada de los algoritmos de solución desarrollados 3](#_Toc67018134)

[1.2. Descripción de las funciones implementadas 3](#_Toc67018135)

[1.3. Descripción de la ejemplificación de las estructuras de datos desarrolladas 4](#_Toc67018136)

[1.4. Problemas sin solución 4](#_Toc67018137)

[1.6. Plan de Actividades realizadas por estudiante 4](#_Toc67018146)

[1.7. Conclusiones 6](#_Toc67018147)

[1.8. Recomendaciones 6](#_Toc67018148)

[1.9. Bibliografía consultada en todo el proyecto 7](#_Toc67018149)

[2. Bitácora 8](#_Toc67018150)

# Breve descripción del proyecto

Donkey Kong Jr es un clásico video juego arcade que consiste en llegar a la cima del mapa utilizando lianas para subir y evitar que los enemigos (kremlings) lo toquen. El objetivo de este proyecto es desarrollar el juego, con la condición crear un servidor en Java para implementar la lógica del juego y crear un cliente en C para controlar el juego.

## Descripción de la utilización de las estructuras de datos desarrolladas.

Para la implementación de este proyecto solo se utilizó la estructura de datos ArrayList, sin embargo, esta estructura ya viene incluida en la librería Java.Util.

Además, se utilizaron dos patrones de diseño:

**Singleton:** Se utilizó el patrón de programación Singleton en la construcción de la clase Server, de tal manera que fuera posible restringir la instanciación de la clase a una instancia "única", y de igual forma realizar su ejecución una única vez.

**MVC:** el modelo vista controlador se utiliza en el servidor para mantenerlo lo más organizado y escalable posible. En este caso, se utilizan tres modelos principales que heredan de Entidad, y son Mono, Fruta y Cocodrilo. En el controlador se utilizan cuatro controladores, uno principal llamado GameManager que es donde se maneja toda la lógica del juego, y un controlador para cada entidad movible; CocodriloController, FrutaController y MonoController. Finalmente, en la vista solo se tiene una Ventana donde se muestra una matriz que representa el juego actual y permite al usuario crear cocodrilos y frutas.

## Descripción detallada de los algoritmos desarrollados.

El juego se implementó utilizando una matriz 100x100, en la se almacenan las entidades del juego. Cada entidad tiene una posición actual y un área, la posición actual es el punto de la matriz donde se ubica la entidad y el área es un arreglo de puntos que representan la silueta del objeto, esta área tiene la función de verificar las colisiones entre las entidades.

El movimiento de las entidades se realiza actualizando la posición actual de cada entidad y automáticamente cada entidad tiene un método abstracto para actualizar su área inmediatamente al moverse. El movimiento del mono se realiza únicamente recibiendo comandos WASD, mientras que los cocodrilos se mueven utilizando un hilo que con un sleep funciona como controlador de velocidad, haciendo que cada vez que se pasa de nivel el sleep disminuya su tiempo.

El salto del mono se implementó utilizando otro hilo y las colisiones del mono, se verifica si el mono está en el suelo comparando los puntos de la posición actual con los puntos de las plataformas y de ser así le permite saltar cada vez que presiona la tecla W, este hilo simplemente le bloquea al jugador la posibilidad de moverse mientras salta y mueve su posición hacia arriba en un intervalo de tiempo y al llegar a una altura máxima empieza a decrementar su posición.

La implementación del uso de las lianas se hace utilizando las colisiones del mono con el entorno, condicionando que si colisiona con una liana el mono deja de caer y tiene la posibilidad de subir en vez de saltar con la tecla W.

## Problemas sin solución

* De manera satisfactoria se logró implementar todo lo necesario según la especificación del proyecto.



## Plan de Actividades realizadas por estudiante

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarea** | **Tiempo estimado** | **Responsable** | **Fecha de entrega** |
|  |  |  |  |

## Problemas encontrados.

* Al trabajar con múltiples hilos en la lógica del juego surgieron muchos problemas de NullPointerException, esto sucedía porque en un principio los campos de la matriz que no pertenecían a una entidad se declaraban como null, y esto ocasionaba choques cuando una entidad cambiaba de posición y dejaba campos de la matriz en null que estaban siendo consultados por otros hilos. La solución a este problema fue utilizar un tipo de entidad Vacio para evitar tratar objetos null y también se utilizaron excepciones para tratar los casos inevitables en los que se intentaba acceder a un campo nulo.
* Otro problema que surgió fue al intentar agregar la librería de allegro en C para la implementación de la interfaz gráfica del cliente, al trabajar con C en Windows suelen encontrarse muchos problemas de compatibilidad dependiendo del IDE, del compilador y otros factores. La solución a este problema fue una exhaustiva investigación además de prueba y error hasta que al final se logró agregar la librería.

## Conclusiones

* El paradigma de la programación orientada a objetos es muy sencillo de entender en comparación a los demás paradigmas ya que se modela como vemos el mundo real.
* El lenguaje de programación C permite un acceso más privilegiado a la memoria de la computadora, permitiendo desarrollar aplicaciones más robustas a costas de una mayor complejidad y una gran curva aprendizaje.
* La herencia en POO es una herramienta muy potente a la hora de reutilizar código, permitiendo resolver problemas en menos líneas de las que se necesitarían en C.

## Recomendaciones

* Leer el manual de usuario antes de ejecutar el programa.
* En la programación orientada a objetos es sumamente importante el uso de patrones de diseño facilitar y estandarizar los proyectos, permitiendo a los desarrolladores acostumbrarse a un estilo de programación conocido por la mayoría.

## Bibliografía consultada en todo el proyecto

chuidiang.org. (2020, 3 21). Socket entre C y java. Retrieved from Socket entre C y java: <http://www.chuidiang.org/java/sockets/cpp_java/cpp_java.php>

# Bitácora

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Estudiante(s)** | **Actividad** |
| 11/04/2021 | Todos | Planeamiento del proyecto y asignación de tareas |
| 13/04/2021 | Todos | Definir e instalar todos los recursos a utilizar en el proyecto (OS, compilador, IDE) |
| 15/04/2021 | Todos | Implementación de los sockets |
| 17/04/2021 | Todos | Desarrollo de la lógica del juego sesión 1 |
| 18/04/2021 | Adrian y Shakime | Desarrollo de la lógica del juego sesión 2 |
| 19/04/2021 | Adrian y José | Desarrollo de la lógica del juego sesión 3 |
| 21/04/2021 | José y Shakime | Implementación de la interfaz en C |
| 25/04/2021 | Todos | Unión del servidor con el cliente |