Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Algoritmos y Estructuras de Datos 1

**Proyecto #1: Invaders**

Profesor: Isaac Ramírez Herrera

Estudiante: Jose Andrés Chavarría Vásquez

Carné: 2017094138

Grupo 1

I Semestre – 2018

**Índice**

1. **Introducción**  3
2. **Descripción del problema** 3
3. **Planificación y administración del proyecto** 4
   1. Features 4
   2. Historias de usuario 4
   3. Distribución por secuencia de uso e importancia 5
   4. Minimal System Span 5
   5. Plan de Iteraciones 6
   6. Descomposición de historias de usuario en tareas 6
4. **Diseño** 7
   1. Diagrama de componentes 7
   2. Diagrama de arquitectura 7
   3. Diagrama de secuencia 8
   4. Diagrama de clases inicial 9
   5. Diagrama de clases final 10
5. **Implementación** 11
   1. Estructuras de datos desarrolladas 11
   2. Algoritmos desarrollados 12
   3. Problemas encontrados 11
6. **Conclusión** 12
7. **Bibliografía** 12
8. **Anexos** 13
9. **Introducción:**

En el juego Invaders, el jugador controla una nave de combate, con el objetivo de destruir la mayor cantidad de filas enemigas antes de ser destruido por las interminables oleadas de enemigos. El jugador debe impedir que los enemigos alcancen la parte inferior de la pantalla o que choquen con la nave del jugador, de no ser así, perderá una vida.

El juego no posee un final, las filas de enemigas son infinitas e incrementa su dificultad con cada una que sea destruida por el jugador. Además, los enemigos cuentan con 6 tipos distintos de formaciones de ataque, en donde pueden realizar movimientos distintos a lo normal, cambiar a su líder en la mitad de un combate e incluso cambiar posiciones con este.

Este juego fue desarrollado en Java, utilizando distintas estructuras de datos para el manejo de las filas enemigas, y se implementaron distintos patrones de diseño para facilitar su programación.

1. **Descripción del problema:**

Invaders es un juego en el que el jugador puede controlar una nave que se mueve horizontalmente en la parte inferior de la pantalla con el fin de eliminar a todos los invasores. Los invasores son filas de enemigos que avanzan de arriba hacia abajo en la pantalla y se mueven de lado a lado. El jugador deberá mover la nave para eliminarlos y evitar que lleguen hacia la parte inferior de la pantalla.

1. **Planificación y administración del proyecto:**
   1. Features:

* Almacenamiento de mejores puntajes con el nombre de cada usuario.
* Nave controlada por el usuario (movimiento y disparo)
* Control de la nave por medio del teclado o con un dispositivo móvil.
* Niveles infinitos.
* Dos tipos diferentes de enemigos (menor y jefe de fila).
* Máximo de vidas para el jugador (si se acaban pierde el juego).
  1. Historias de usuario:

1. Como jugador, quiero controlar una nave para luchar contra otras naves enemigas.
2. Como jugador, quiero que mi puntuación se guarde en una lista de mejores puntuaciones locales, para aumentar la competitividad.
3. Como jugador, quiero sumar puntos al destruir naves enemigas y progresar en la partida.
4. Como jugador, quiero que los enemigos tengan una serie de distintos movimientos o capacidades especiales para evitar la monotonía.
5. Como jugador, quiero que, al ir progresando en la partida, esta se vuelva más desafiante.
6. Como jugador, quiero que la nave tenga un máximo de vidas, para perder el juego en caso de que falle en destruir las naves a tiempo.
7. Como jugador, quiero poder controlar la nave con un dispositivo móvil, como un celular, para tener otras opciones de control de esta.
   1. Distribución por secuencia de uso e importancia:

Secuencia de uso

Destrucción de enemigos y jugador Importancia: Media

Características especiales de las hileras Importancia: Media

Control desde dispositivo móvil Importancia: Baja

Movimiento y disparos del jugador Importancia: Alta

Generar hileras de enemigos Importancia: Alta

Generar naves (enemigos y jugador) Importancia: alta

**Ejecución**

**Final**

**Inicialización**

Importancia

Tabla de puntuaciones Importancia: Baja

**Minimal System Span**

* 1. Minimal System Span:

Como se muestra en el diagrama anterior, el “minimal system span”, o sea, las características mínimas para tener un juego funcional, incluyen los features:

* Generar el jugador.
* Generar enemigos.
* Generar las hileras de enemigos.
* Movimiento y disparos del jugador.

Con las funciones anteriores, se tiene un juego funcional que el jugador puede utilizar, aunque no se hayan implementado el resto de funciones. Es como la versión alpha del desarrollo del juego, es la primera versión.

* 1. Plan de Iteraciones:

Secuencia de uso

Movimiento y disparos del jugador Importancia: Alta

Generar hileras de enemigos Importancia: Alta

**Ejecución**

**Final**

**Inicialización**

Importancia

Generar naves (enemigos y jugador) Importancia: alta

**1. Minimal System Span**

Destrucción de enemigos y jugador Importancia: Media

Características especiales de las hileras Importancia: Media

**2. Iteración**

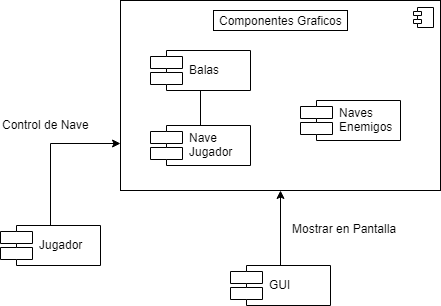
Control desde dispositivo móvil Importancia: Baja

Tabla de puntuaciones Importancia: Baja

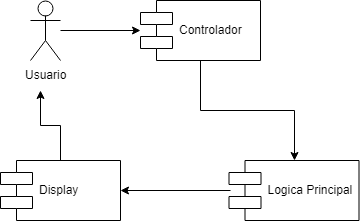
**3. Iteración**

* 1. Descomposición de Historias de Usuario en tareas:

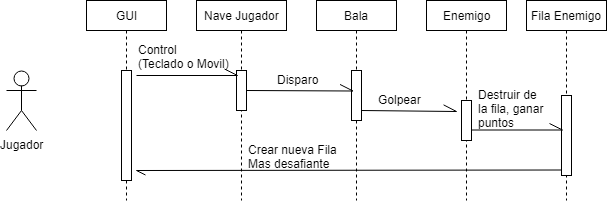
1. Ventana de puntuaciones máximas. Almacenamiento de datos.
2. Manejo, destrucción y disparo de la nave jugador.
3. Movimiento y destrucción de naves enemigas.
4. Aumento de la dificultad.
5. Vida máxima del jugador y pantalla de Game Over.
6. Control de la nave desde un celular.
7. **Diseño:**
   1. Diagrama de Componentes:



* 1. Diagrama de Arquitectura:

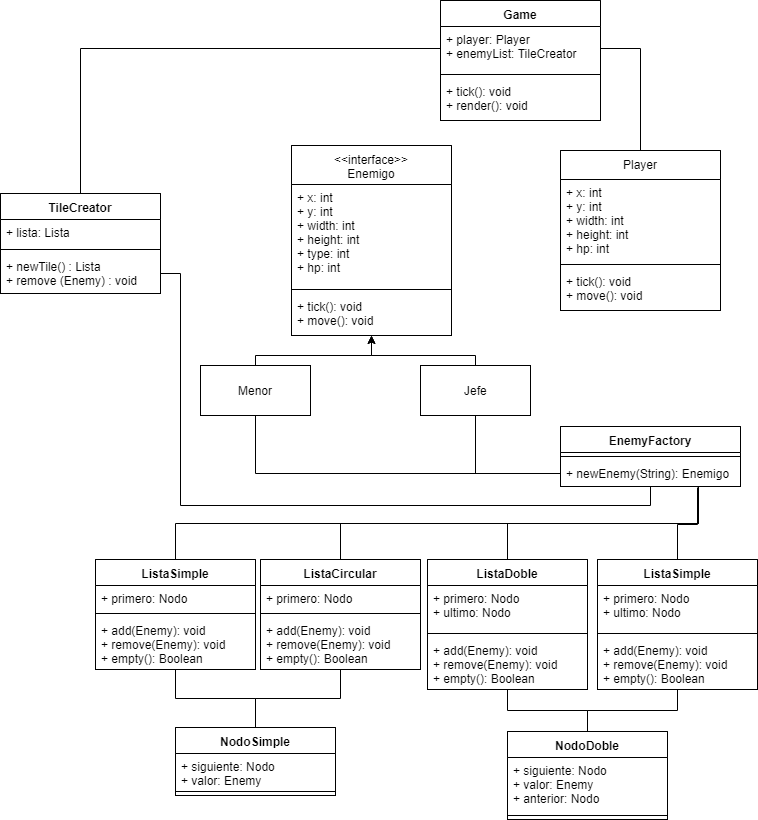


* 1. Diagrama de Secuencia:

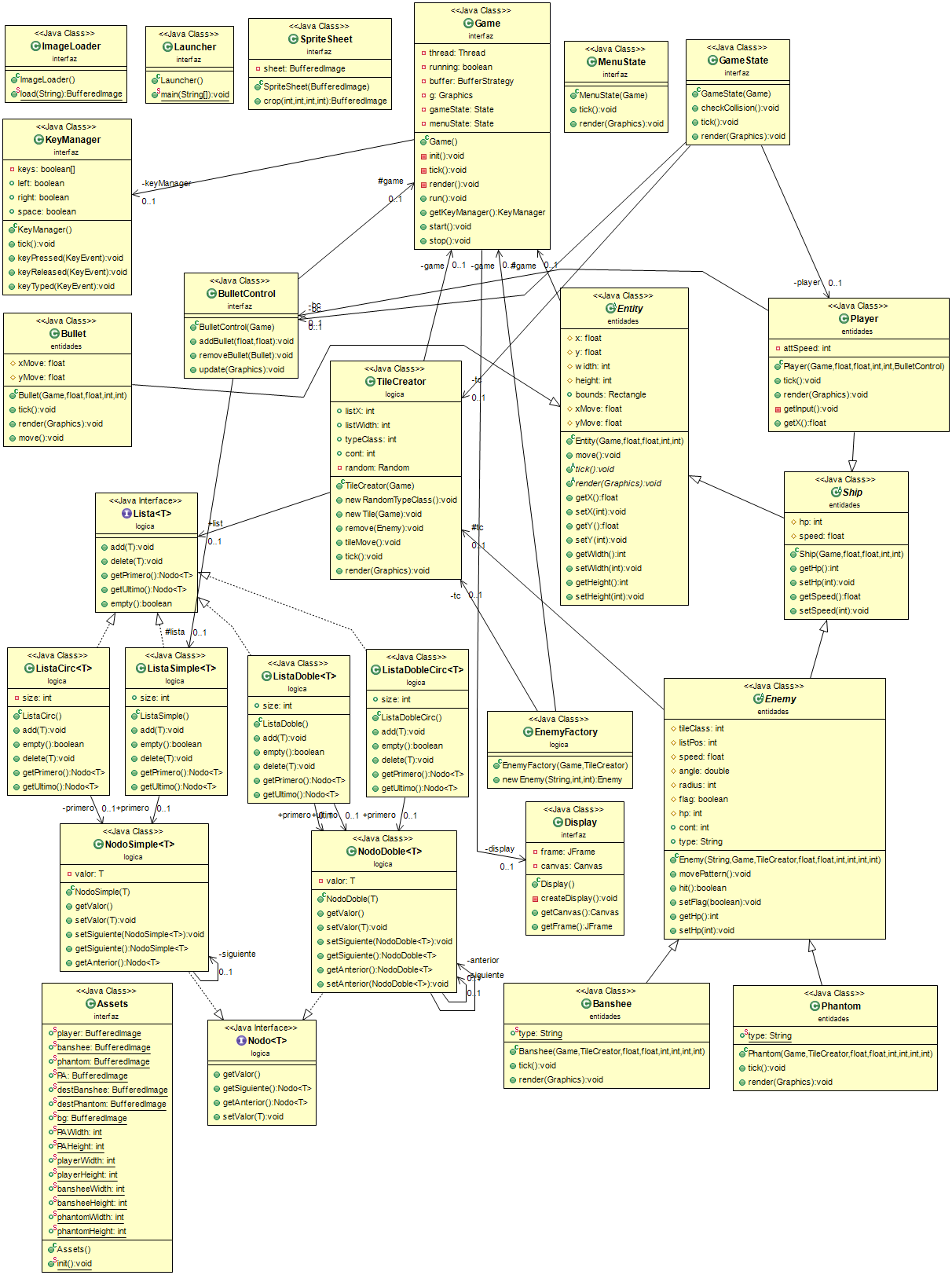


Se seleccionaron las historias de usuario “a”, “b”, “e” y “g”, para realizar este simple diagrama de secuencia.

* 1. Diagrama de Clases inicial:



* 1. Diagrama de Clases final:



1. **Implementación:**
   1. Estructuras de datos desarrolladas:
2. Lista enlazada simple: Es una estructura similar a un arreglo, con la ventaja de poder agregar datos a esta, sin necesidad de crear una lista totalmente nueva cada vez. Utiliza Nodos para almacenar estos datos. Cada nodo tiene una referencia al nodo siguiente, el ultimo nodo de la lista apunta a null.
3. Lista doblemente enlazada: Es igual a la lista simple, con la excepción de que sus nodos tienen referencias a su nodo siguiente y nodo anterior.
4. Lista enlazada circular: Es una variación de la lista simple, su diferencia es en su nodo final, ya que este no apunta a null, si no que apunta al primer nodo de la lista.
5. Lista circular doblemente enlazada: Es una lista circular con las propiedades de los nodos de la lista doblemente enlazada. Su primer nodo también apunta al nodo final.
   1. Algoritmos desarrollados:
6. Crear filas de enemigos: Es un método de la clase TileCreator, este se encarga de crear una nueva fila de enemigos. Primero crea un numero aleatorio, el cual usa para crear una nueva lista dependiendo del tipo de fila. Luego utiliza el patrón de diseño “Factory” para crear los enemigos y los coloca en pantalla.
7. Control de colisiones: Este algoritmo se encarga de revisar que cada bala no toque ningún enemigo, de ser así, la bala desaparece y el enemigo pierde puntos de vida. Además, revisa que el jugador no choque con ningún enemigo, o este perderá una vida.
8. Movimiento de enemigos: Mueve a la fila como una sola entidad, utilizando la posición en el eje X del primero y ultimo enemigo de la fila, cuando estos valores exceden un punto, toda la fila cambia de dirección y se mueve en sentido contrario. Además, si la fila es de tipo “E”, los enemigos realizan movimientos circulares utilizando las funciones matemáticas seno y coseno para definir su posición en el eje X, Y.
9. Controlador: Revisa cuales teclas el usuario está tocando, mueve la nave del jugador en la dirección especificada y permite disparar.
   1. Problemas encontrados:

* Al utilizar distintos tipos de listas para cada tipo de fila de enemigo, el método que crea la fila debe retornar una lista. Al ser todas distintas, se crean las interfaces Lista<T> y Nodo<T> para poder retornar cualquier tipo de lista sin problemas.
* Al crear ciertos tipos de filas de enemigos, el generador de fila no logra crear el numero completo de naves deseadas, además de que, al destruir la fila, no se genera una nueva. La solución aun esta en progreso.

1. **Conclusiones:**

Se logró la implementación de listas enlazadas y sus variaciones para el control de las filas enemigas. Se aplicaron patrones de diseño para facilitar el trabajo al programar, además de ser buenas prácticas de programación. Se investigó acerca de cómo desarrollar una aplicación en Java, además de la investigación necesaria para entender la programación orientada a objetos utilizada en Java.

1. **Bibliografía:**

Java 2D games tutorial. Recuperado de: zetcode.com/tutorials/javagamestutorial/

1. **Anexos:**

Git hub: <https://github.com/JoseChV/Invaders>