



INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN
ÁREA ACADÉMICA DE INGENIERÍA EN COMPUTADORES
ALGORITMOS Y ESTRUCTURAS DE DATOS I (CE-1103)

Proyecto 1
Invaders

Realizado por:
EDUARDO MOYA BELLO - 2015096664

Profesor:
Ing. Isaac Ramírez, M.SI.

Cartago, 4 de abril de 2017

Índice

Introducción	3
Descripción del problema	4
Planificación y Administración del Proyecto	5
Features e historias de usuario	5
Features	5
Historias de usuario	5
Distribución de historias de usuario por criticalidad y secuencia de uso	6
Minimal system span	7
Plan de iteraciones	7
Descomposición de cada user story en tareas	9
Interfaz gráfica y jugador	9
Control por teclado	9
Sistema de disparos	9
Movimiento en filas de enemigos utilizando listas enlazadas	9
Seis tipos de filas diferentes	10
Control por celular	10
Mostrar la información en el celular	10
Diseño	11
Diagrama de componentes	11
Diagrama de secuencia	11
Interfaz gráfica y jugador	11
Control por teclado	13
Sistema de disparos	13
Movimiento en filas de enemigos utilizando listas enlazadas	16
Seis tipos de filas diferentes	17
Control por celular	17
Diagrama de clases inicial	18
Diagrama de clases final (generado utilizando el IDE)	19
Implementación	20
Estructuras de datos desarrolladas	20
Listas enlazadas	20
Otros	20
Algoritmos desarrollados	20
Problemas encontrados	20
Uso de Git y del lenguaje Java	20

Desconocimiento de alguna interfaz gráfica en Java	21
Inexperiencia en programación orientada a objetos	21
Desarrollo del socket y creación de la aplicación de Android	21
Conclusiones	23
Referencias bibliográficas	24
Anexos	25
Anexo 1: tipos de filas de enemigos	25
Anexo 2: diagramas de secuencia para las filas de enemigos clase A, B, C y D	26
Clase A	26
Clase B	27
Clase C	28
Clase D	29

Introducción

La Programación Orientada a Objetos así como los patrones de diseño, son muy importantes en la programación, ya que se utilizan en gran variedad de aplicaciones por su nivel de confianza y eficiencia. Para el presente caso, se aplicarán dichos paradigmas en la elaboración de un videojuego.

La idea del presente proyecto es diseñar e implementar el juego al estilo de Space Invaders, utilizando listas enlazadas. Esto se logrará implementando listas enlazadas y algunas de sus variaciones, investigando acerca del desarrollo de aplicaciones en el lenguaje Java, investigando acerca de la programación orientada a objetos en Java y aplicando patrones de diseño. Al final, se pretende dominar dichos aspectos para el curso de Algoritmos y Estructuras de Datos I.

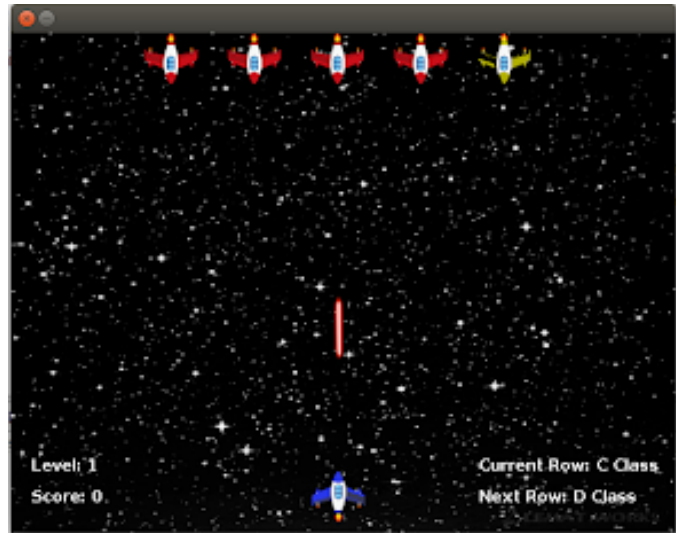
Para la elaboración del juego se utilizará el entorno de desarrollo integrado (IDE) Eclipse, así como Android Studio. La presente documentación se estructura de la siguiente manera: primero se describe el problema a resolver en el desarrollo del juego, posteriormente se indica la manera en la que se planificó y administró (principalmente tomando en cuenta historias de usuario), después se mostrará el diseño del proyecto mediante diferentes diagramas y finalmente se hablará acerca de la implementación.

Cabe destacar que se también se concluirán los principales aprendizajes del proyecto y se brindará toda la información bibliográfica necesaria, para respaldar lo aplicado. Al final del documento, se cuenta con dos anexos que complementan algunos aspectos del diseño del proyecto.

Cabe destacar que se manejará un historial de versiones de código (para este caso Github), con el objetivo de mostrar el avance del proyecto y brindar de manera abierta, acceso al código.

Descripción del problema

Como se mencionó anteriormente, para el presente trabajo se debe crear un juego de naves espaciales basado en el juego Space Invaders. La idea es que el jugador, mediante el teclado o el celular pueda mover una nave, la cual se mueve de forma horizontal en la parte inferior de la pantalla.



Los enemigos (invasores) se mueven según el tipo de fila, sin embargo, siempre tienden a acercarse al jugador, por lo que el objetivo es eliminar a los enemigos. Esto se logra mediante disparos. Para el desarrollo de esta aplicación, se decidió también darle al jugador la libertad de elegir en qué momento disparar.

Los tipos de filas de enemigos están previamente definidos en el enunciado del proyecto, y se encuentran disponibles en el anexo 1. Estas hileras se implementan mediante listas enlazadas. También, se requiere que las filas se muevan más rápidamente conforme avanzan los niveles. Asimismo, los enemigos deben moverse hacia al centro cuando otro es eliminado.

En la pantalla, es requisito ver el tipo de hilera actual y el siguiente. Además, se muestra el nivel actual y la cantidad de puntos que ha acumulado el jugador.

Con respecto a los métodos de entrada, además del teclado se debe poder jugar con el celular. Para esto, se utilizará el acelerómetro, el cual determinará la dirección de movimiento del jugador. En el celular también debe verse la información de la pantalla indicada anteriormente.

Planificación y Administración del Proyecto

Features e historias de usuario

Features

Las características principales que tendrá el juego son las siguientes:

1. Se contará con una interfaz gráfica que muestre los siguientes aspectos: nave, fila de enemigos, nivel, puntuación, clase de fila de enemigos actual y clase de fila de enemigos siguiente.
2. Los invasores que se moverán en fila completa, de lado a lado y hacia abajo de la pantalla, mediante la creación de listas enlazadas.
3. Los enemigos estarán dispuestos en seis diferentes clases de filas, especificadas en el enunciado.
4. Se dispondrá de un sistema de disparos para que el jugador sea capaz de eliminar a los enemigos.
5. Se podrá controlar una nave horizontalmente utilizando el teclado.
6. Se podrá controlar una nave horizontalmente utilizando el acelerómetro de un celular.
7. En el celular también estarán disponibles los aspectos de la feature 1.

Historias de usuario

Las estadísticas del juego mediante historias de usuario se pueden observar en la siguiente tabla

Tema	Como	Quiero	Para	Notas	Prioridad	Status
Control por teclado	Player	Poder mover y disparar con la nave usando el teclado	Interactuar con el juego	Con flechas y la tecla espacio	Necesario	

Tema	Como	Quiero	Para	Notas	Prioridad	Status
Listas enlazadas	Developer	Utilizar estructuras de datos	Aumentar eficiencia	Sin utilizar las listas enlazadas de Java	Necesario	
Disparos	Player	Eliminar enemigos	Avanzar en el juego	Los disparos no deben ser totalmente seguidos	Necesario	
Seis tipos de filas de enemigos diferentes	Player	Jugar varios modos de juego	Mejorar la experiencia	Seis filas diferentes indicadas en el enunciado	Necesario	
Control con el celular	Player	Moverme y disparar con el celular	Mejorar la experiencia			
Información en la interfaz gráfica	Player	Poder observar el juego representado gráficamente y poder ver mi progreso, la fila actual y la siguiente	Visualizar mi desempeño en el juego			
Información en la aplicación del celular	Player	Poder ver mi progreso, la fila actual y la siguiente en mi celular	Estar informado			

Distribución de historias de usuario por criticalidad y secuencia de uso

A continuación, se mostrará una tabla con las historias de usuario. Verticalmente, se ha ordenado las historias de usuario según su criticidad, donde 5 es la mayor y 1 es la menor. Por otra parte, la secuencia de uso se puede observar de manera horizontal, donde las primeras en la secuencia se encuentran del lado izquierdo.

↑ Criticidad ↑	5	Interfaz gráfica	Control por teclado	Sistema de disparos donde el jugador sea capaz de eliminar a los enemigos	Movimiento en fila de enemigos por listas enlazadas			
	4					Seis tipos diferentes de filas		
	3							
	2						Control por celular	
	1							Mostrar información en pantalla del celular
		→ Secuencia de uso →						

Minimal system span

Para que el juego sea funcional, requiere que se cumplan como mínimo las siguientes condiciones:

- Interfaz gráfica.
- Control por teclado.
- Sistema de disparos donde el jugador sea capaz de eliminar a los enemigos.
- Movimiento en fila de enemigos.

Plan de iteraciones

El plan inicial de iteraciones se muestra a continuación. Las iteraciones finales se encuentran disponibles en los commits del repositorio de Github adjunto en la bibliografía. En los problemas se hablará de algunas razones por las que no fue posible cumplir las fechas límites establecidas.

Historia de usuario	Iteración	Asunto	Fecha límite preferible
Interfaz gráfica	1	Creación del proyecto. Vinculación con Git.	19 de marzo

Historia de usuario	Iteración	Asunto	Fecha límite preferible
Interfaz gráfica	2	Definir los parámetros iniciales de la ventana y frame del juego	19 de marzo
Interfaz gráfica	3	Creación de la clase Player	20 de marzo
Control por teclado	4	Soporte para teclado, de manera que el jugador se pueda mover con las flechas.	21 de marzo
Sistema de disparos	5	Implementación de balas y posibilidad de crearlas mediante el teclado.	23 de marzo
Sistema de disparos y filas de enemigos	6	Implementación de listas enlazadas y todos sus métodos	25 de marzo
Seis tipos diferentes de filas	7	Creación de enemigos y filas de enemigos	27 de marzo
Filas de enemigos	8	Desarrollo del algoritmo para la colisión de las balas con los enemigos	1 de abril
Control por celular	9	Desarrollo de una sencilla aplicación para Android	2 de abril
Control por celular y mostrar información en el celular	10	Programación de un socket para comunicar el celular con la computadora	5 de abril

Descomposición de cada user story en tareas

Las historias de usuario mencionadas anteriormente, requieren de varias tareas para completarse exitosamente. El repositorio de Github disponible en las referencias bibliográficas, permite observar el avance de dichas tareas mediante commits.

Interfaz gráfica y jugador

1. Determinar e implementar las características básicas de la ventana así como del o de los frames a utilizar.
2. Configurar la lógica básica del juego, que considere el puntaje, el nivel y los tipos de filas de enemigos (aunque no estén implementados todavía).
3. Aplicar un fondo de pantalla.
4. Desarrollar un objeto tipo jugador y mostrarlo en la interfaz gráfica.

Control por teclado

1. Crear una clase que detecte las teclas presionadas.
2. Notificar al jugador si se ha tocado una tecla para mover dicho objeto.

Sistema de disparos

1. Crear una clase Bullet (bala) que cree una bala cuando la persona presione la tecla de disparo.
2. Mostrar la bala en la interfaz.
3. Implementar una lista de balas para todas las balas que se hayan disparado.
 - a. Esta tarea requiere de la creación de listas enlazadas.
4. Eliminar todas las balas que hayan colisionado (para cuando se creen los enemigos) o salido del área mostrada en la interfaz.

Movimiento en filas de enemigos utilizando listas enlazadas

1. Crear una clase Enemy (enemigo).
2. Crear una clase abstracta EnemyList (lista de enemigos), que contenga los métodos en común de todas las clases de filas de enemigos.

3. Administrar las colisiones entre balas y cada enemigo de la lista de enemigos.

Seis tipos de filas diferentes

1. Implementar cada una de las seis clases de fila, donde se desarrolle la lógica específica de los enemigos.
2. Para algunas clases de filas será necesaria la creación de un objeto Boss (jefe) el cual heredará de Enemy.
3. Utilizar el patrón de diseño Factory para la implementación de las filas, según su clase.

Control por celular

1. Desde el celular, se debe implementar una aplicación que permita disparar y moverse.
2. Para el juego, implementar un servidor para recibir la información del celular.
3. Utilizar la información recibida por el celular para moverse y disparar.

Mostrar la información en el celular

1. Implementar un servidor en el celular para recibir los datos del juego.
2. Implementar un cliente en el juego, para que envíe la información del juego.

Diseño

Diagrama de componentes

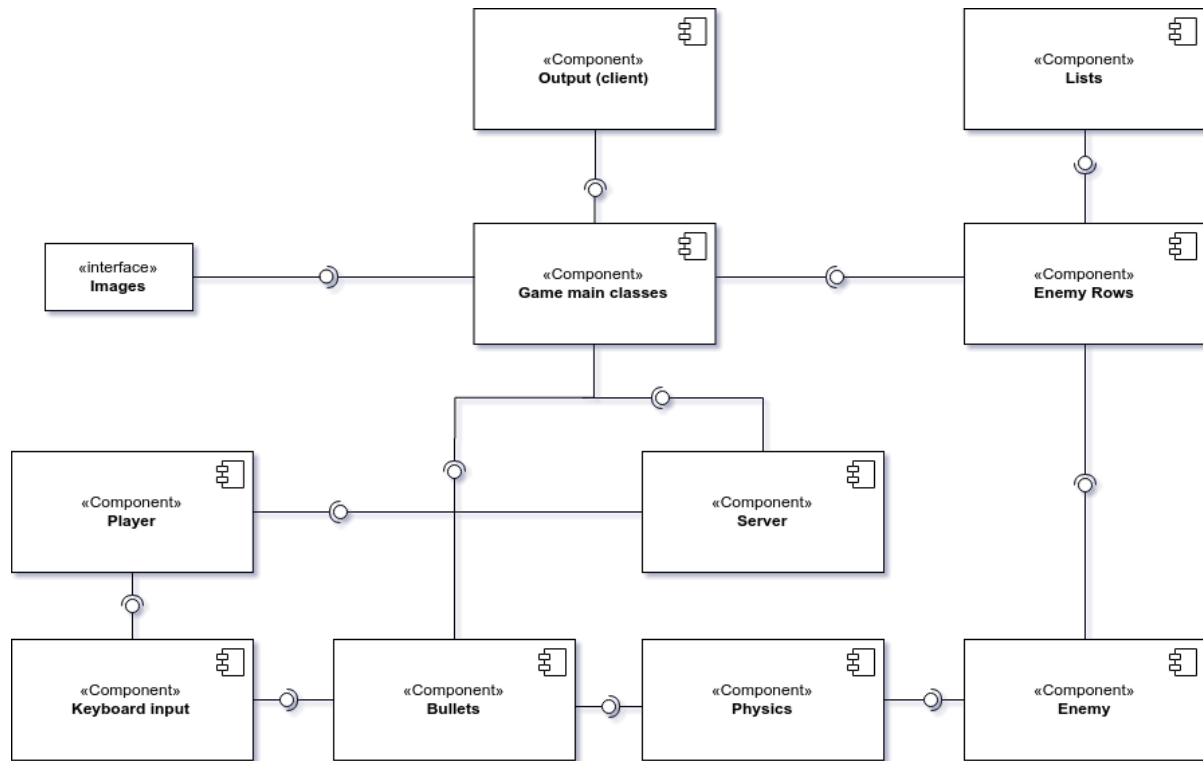
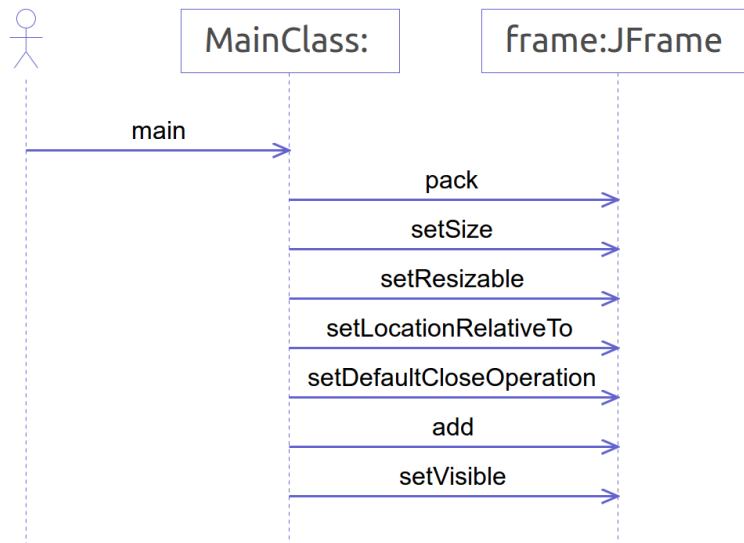


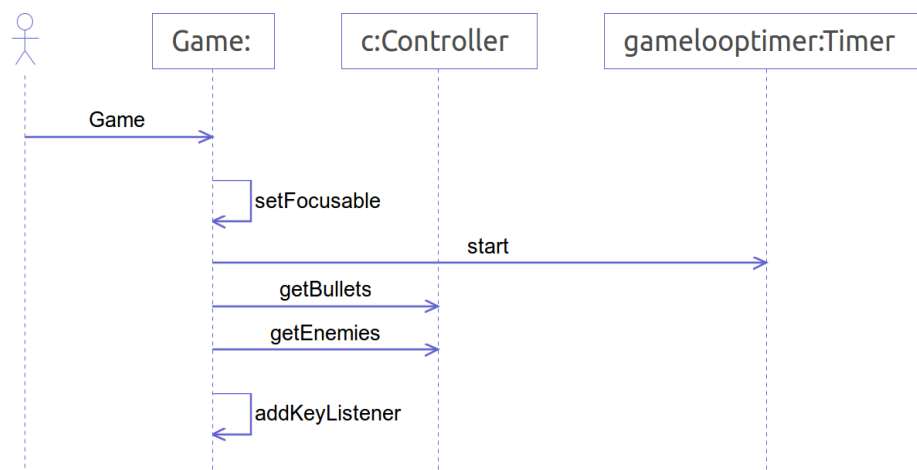
Diagrama de secuencia

Interfaz gráfica y jugador

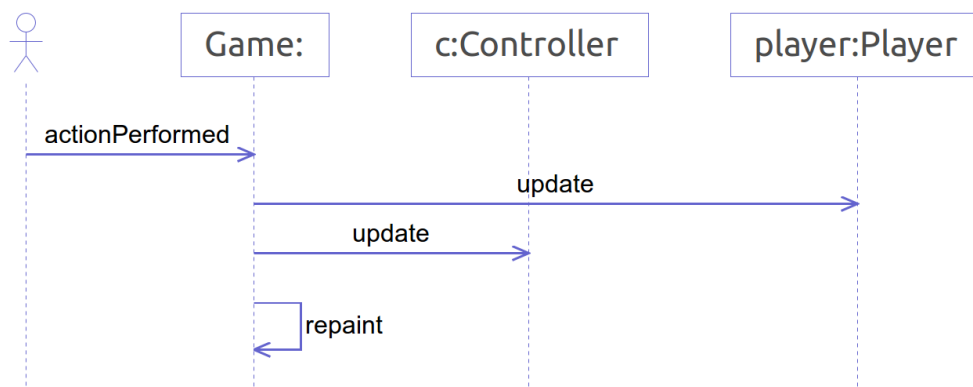
Para la elaboración de la interfaz gráfica, primero se definen los parámetros iniciales de la ventana y el frame.



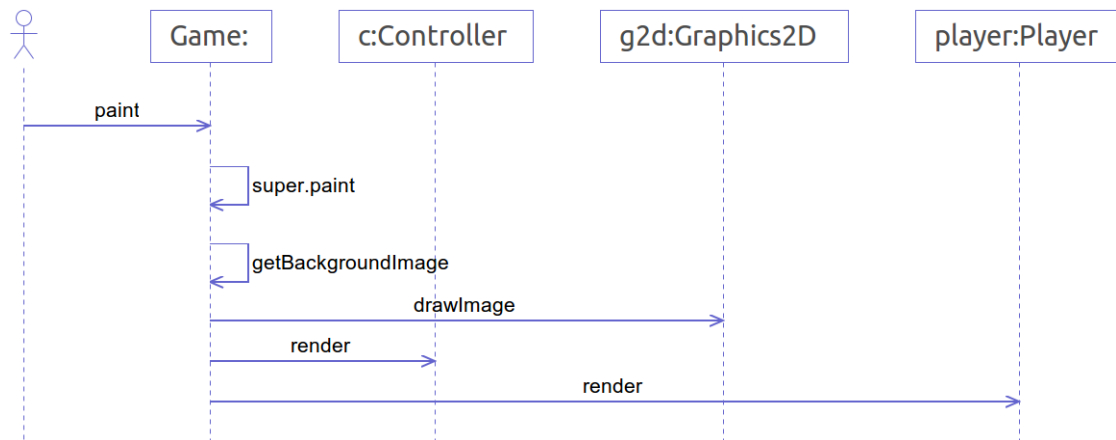
Posteriormente, el método add, inicializa el juego mediante la clase Game. Esto también inicializa el jugador.



Posteriormente, se inicia el loop del juego:

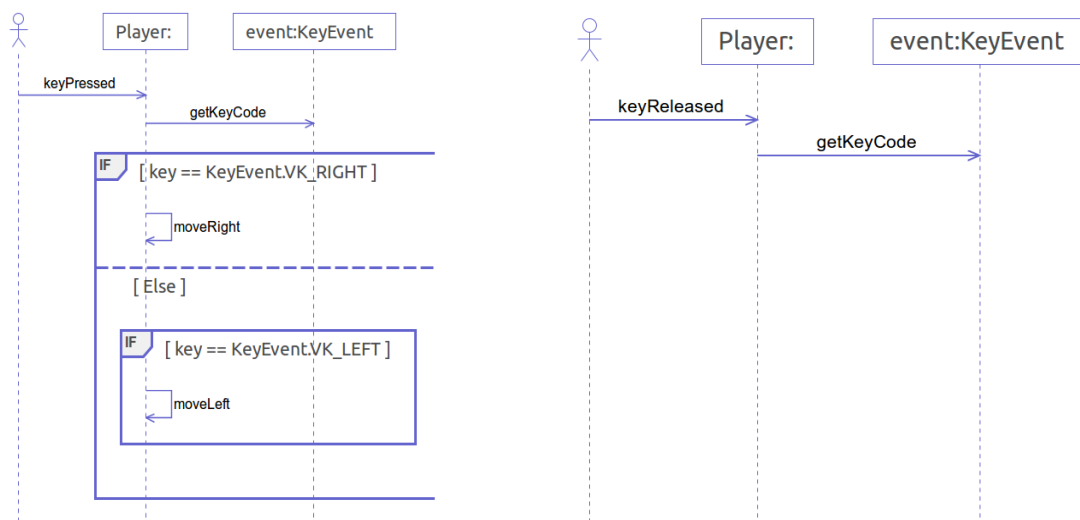


Y también se renderizan las imágenes y textos en el frame:



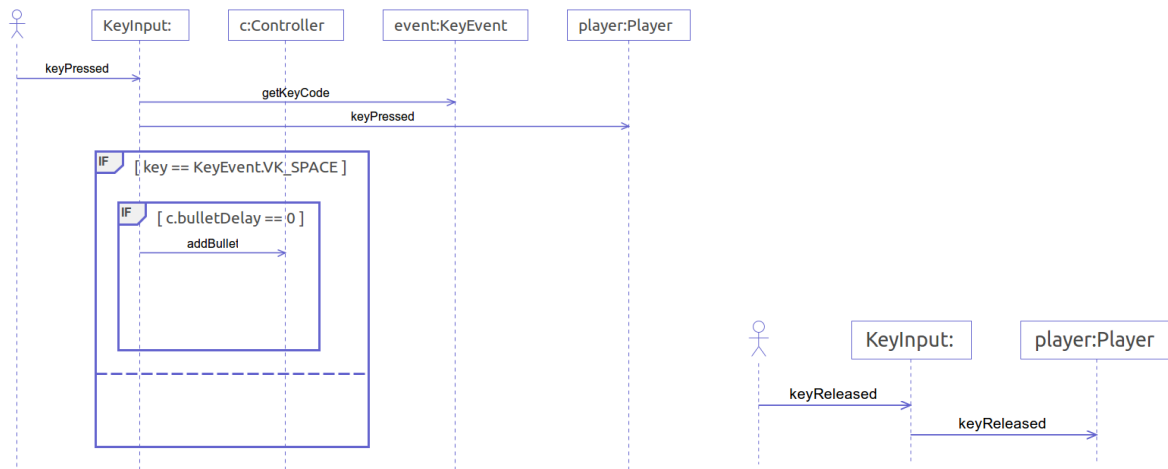
Control por teclado

Hay dos partes en el código donde se controla el juego por el teclado. En la clase `Player` propiamente, se define el movimiento del jugador:

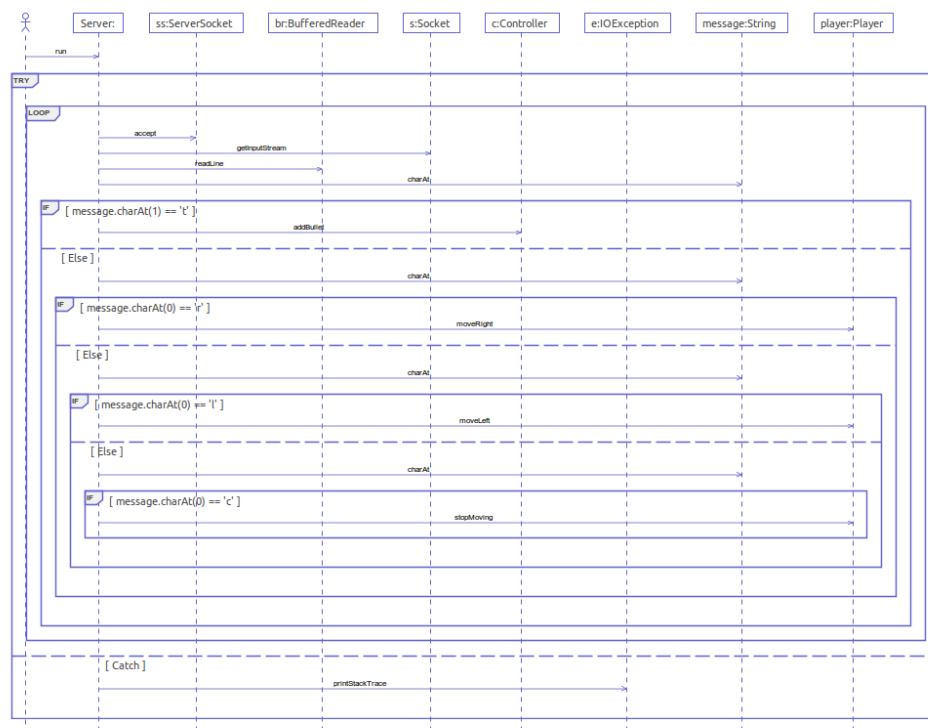


Sistema de disparos

El sistema de disparos funciona con la clase `Bullet`. Primero se utiliza la entrada (teclado o celular) para crear una bala y agregarla a una lista enlazada de balas. Posteriormente, se valida constantemente si cada bala chocó con algún enemigo. Cuando se inicializa el controlador, se crea la lista de balas. Con ayuda de la clase `KeyInput`, se considera la creación de las balas con la tecla espacio:

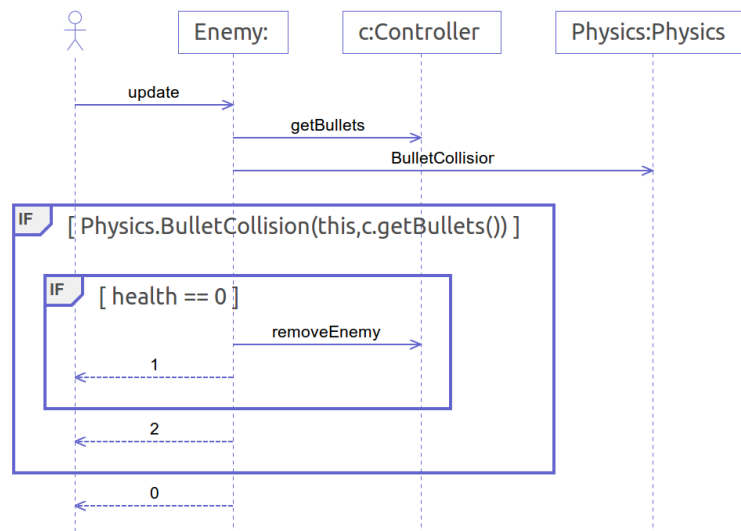
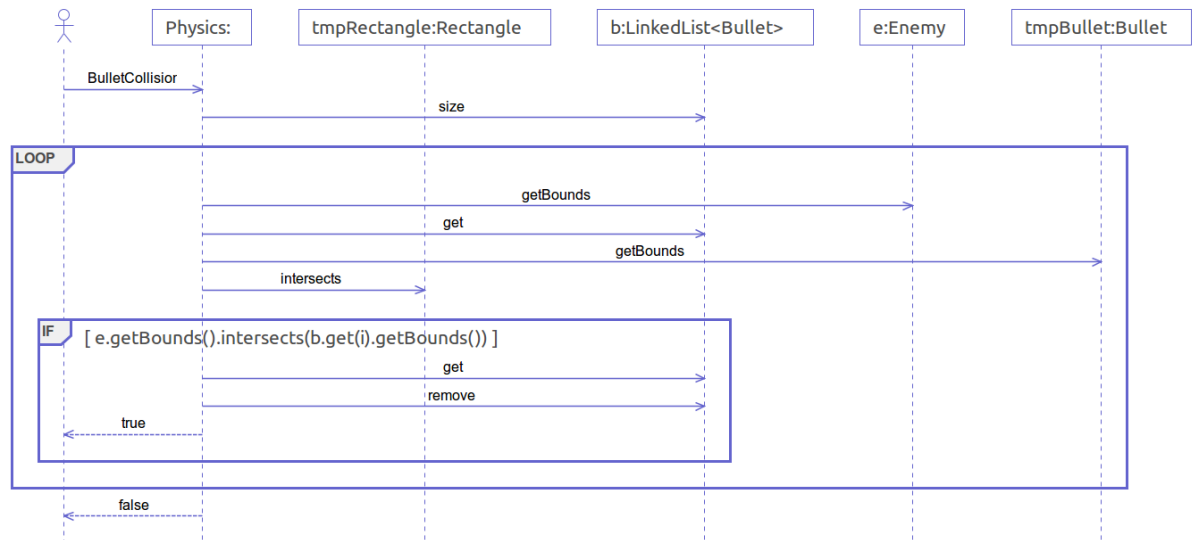


En caso de que la entrada sea el celular, la secuencia es la siguiente:

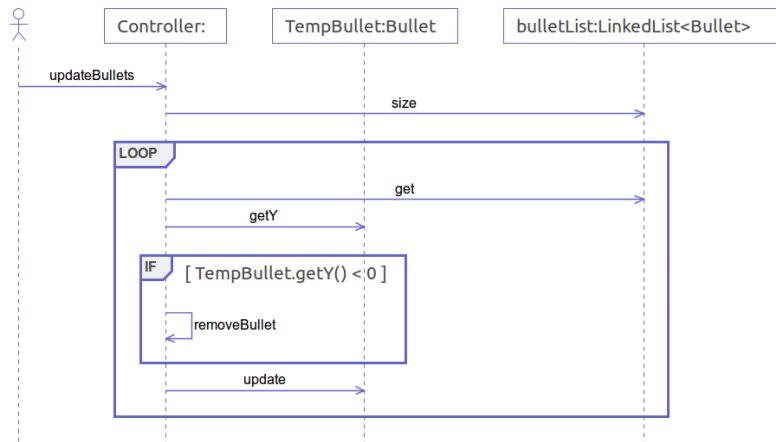


Se debe notar la primer condición. Si el String recibido en la posición 1 tiene una t, significa que el disparo enviado por el celular es un true, lo que crea una bala.

Para validar si la bala ha chocado, se aprovecha que cada bala y enemigo tienen un rectángulo asociado. La clase Rectangle tiene un método que permite saber si dos rectángulos chocaron, lo que simplifica la validación. Esto se consigue con ayuda de la clase Physics:

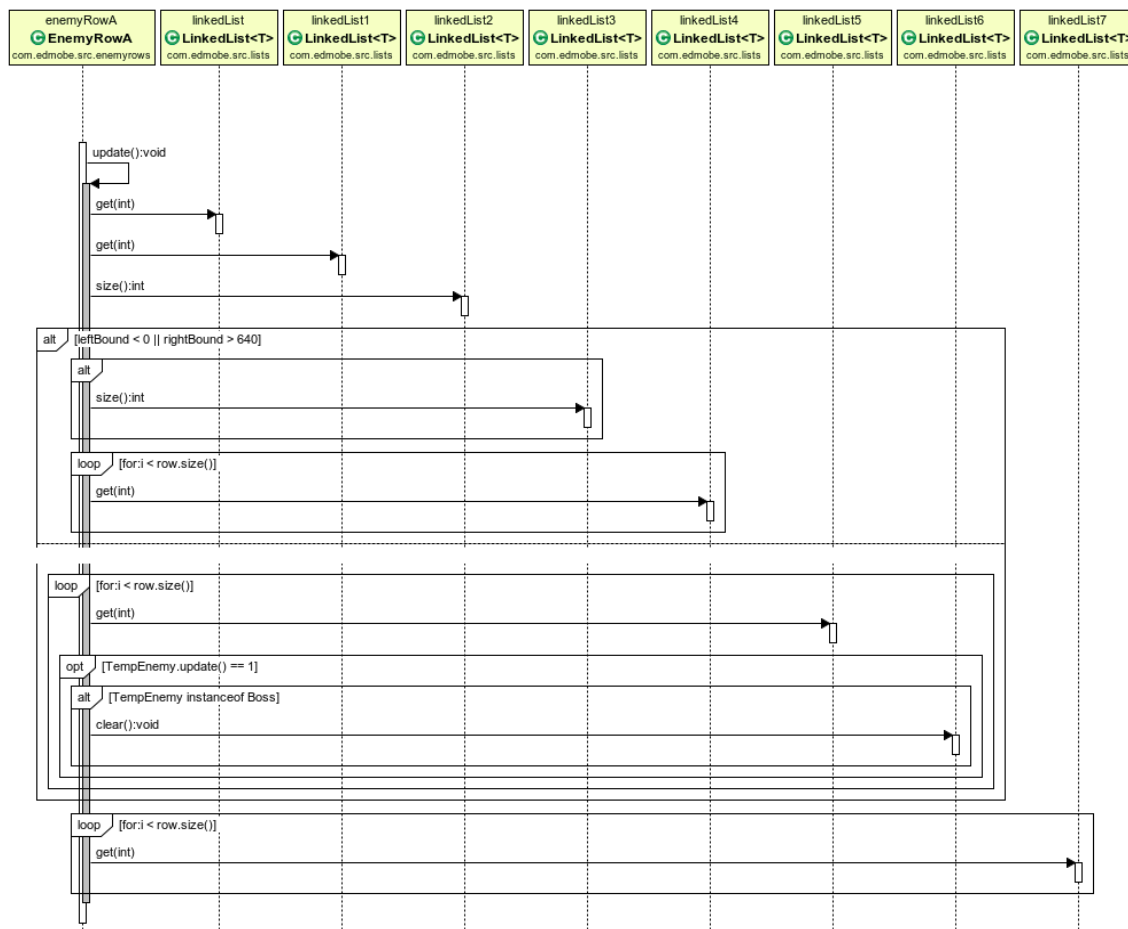


Cabe destacar que si las balas chocaron o salen del rango de visibilidad, se eliminan:

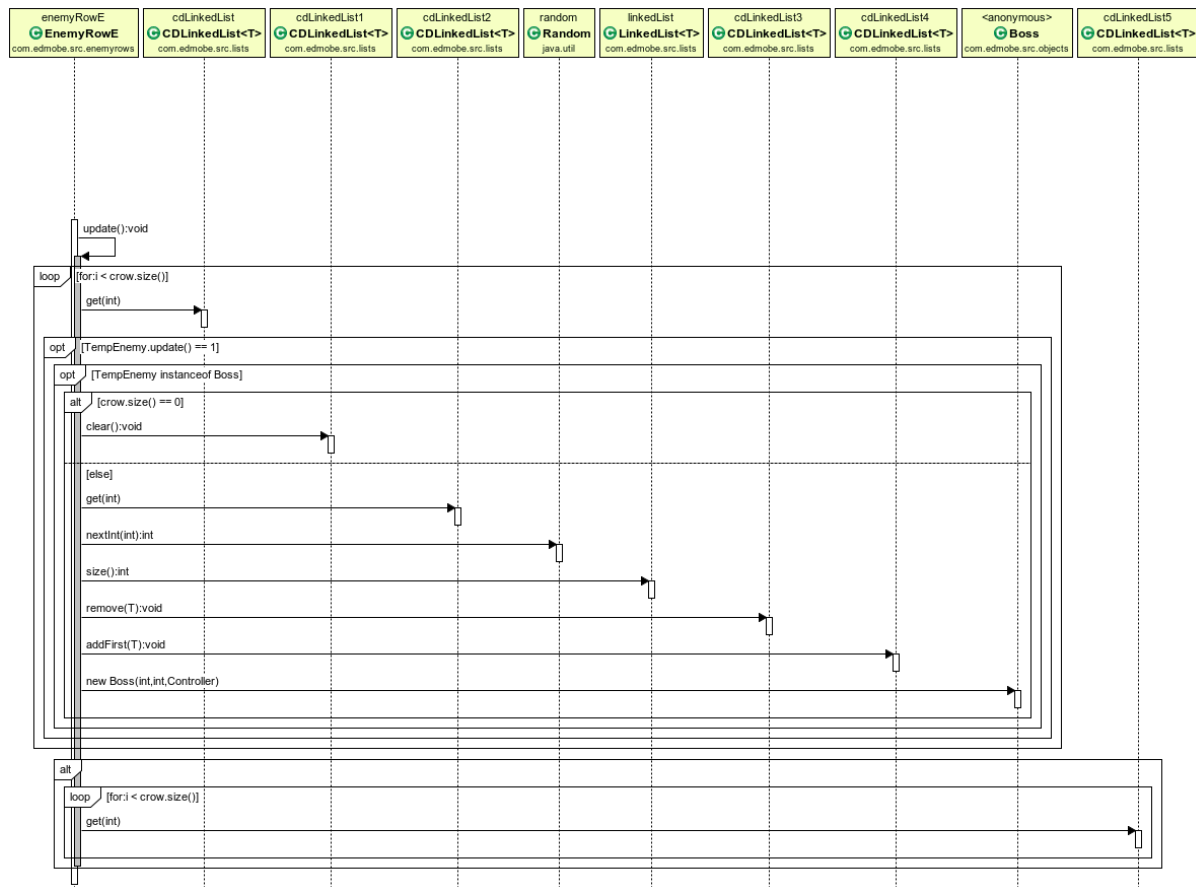


Movimiento en filas de enemigos utilizando listas enlazadas

Hay dos tipos de movimiento de enemigos. El primero es de lado a lado y bajando en cada extremo. El segundo (presente en la hilera Clase E) es igual bajando, pero el movimiento de enemigos en sentido de las manecillas del reloj respecto al jefe. El primer movimiento se implementa de la siguiente manera:



Y el movimiento circular se implementa como procede:



Seis tipos de filas diferentes

Se crearon seis diferentes clases de fila cuyo comportamiento es distinto y se define según las indicaciones del anexo 1. En los dos diagramas anteriores se observaron la Clase Basic y la Clase E. Las clases A, B, C y D se muestran respectivamente en el anexo 2.

Control por celular

Como ya se mencionó, se implementó un servidor y un cliente para el envío y recepción de información. La secuencia del servidor se implementa según el segundo diagrama del Sistema de Disparos. Finalmente, los mensajes se envían como procede:

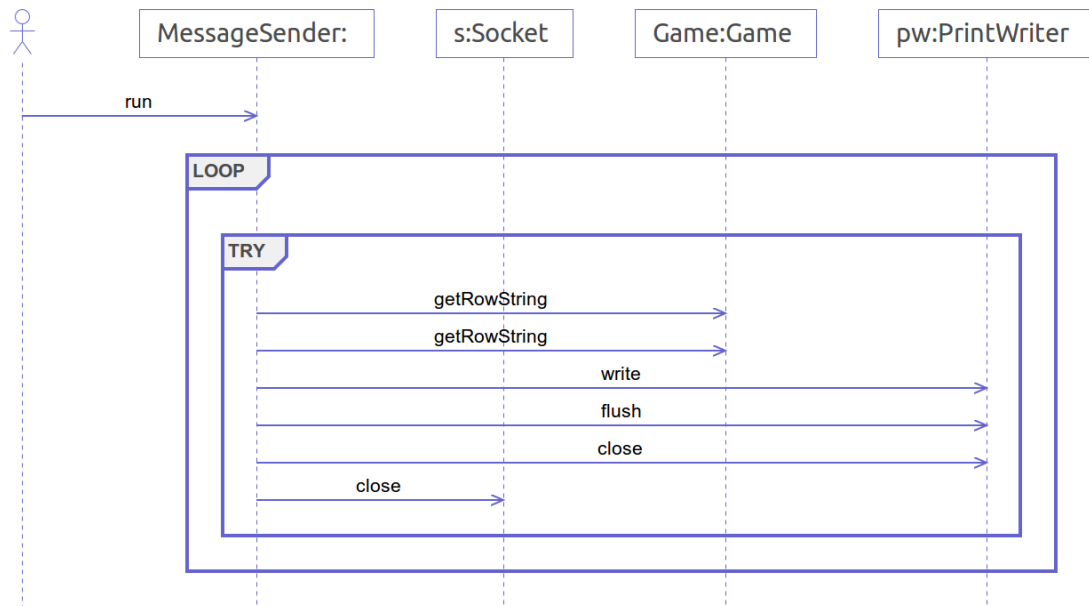


Diagrama de clases inicial

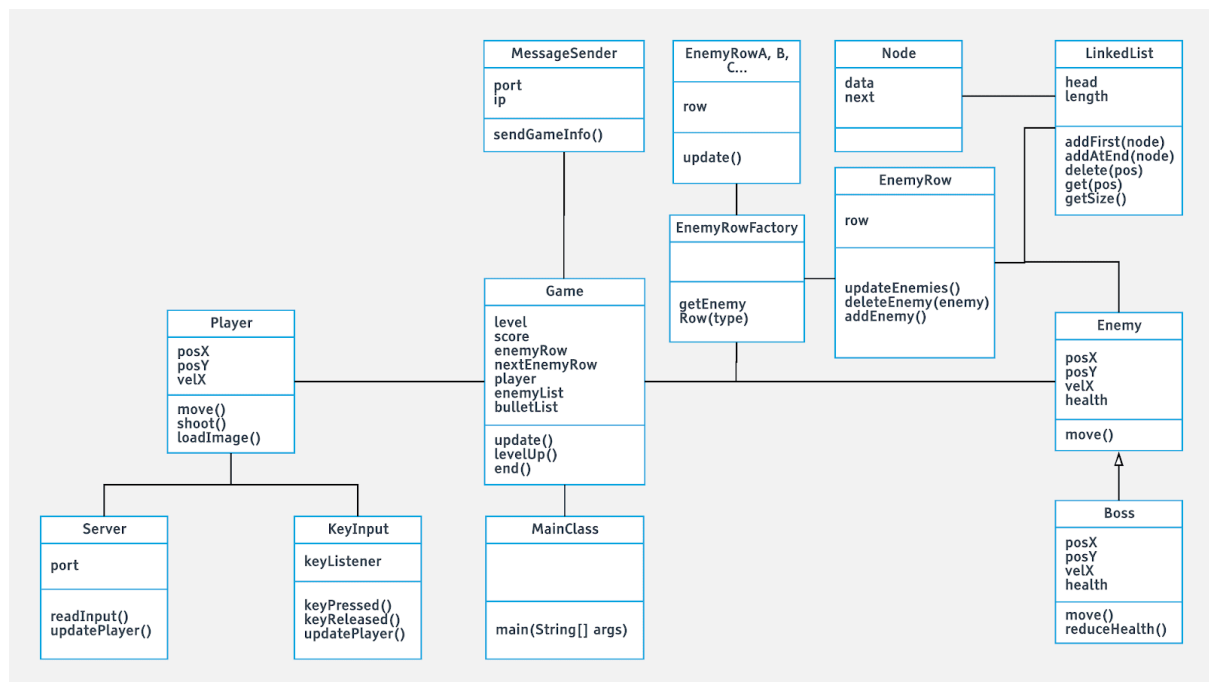
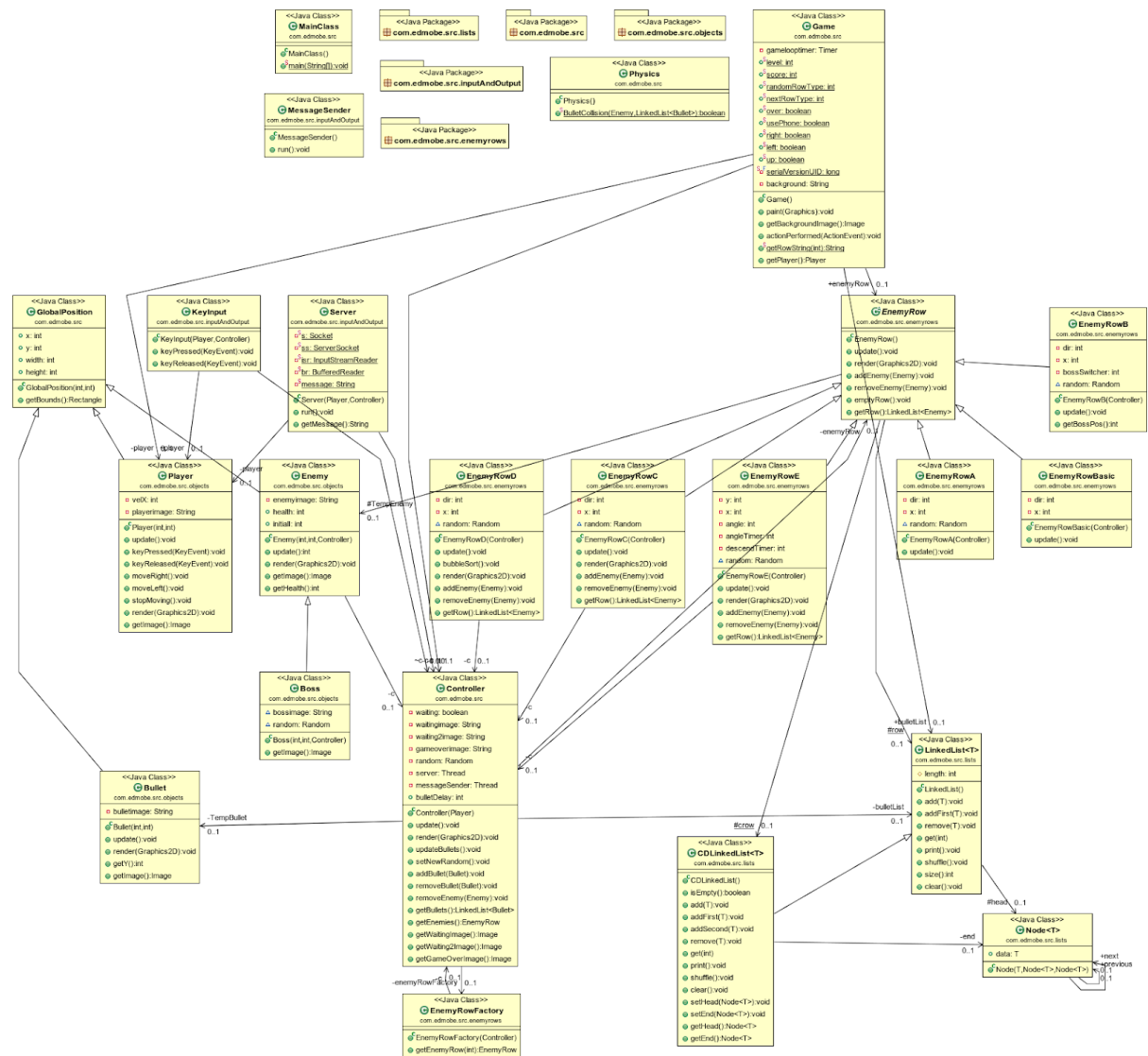


Diagrama de clases final (generado utilizando el IDE)



Implementación

Estructuras de datos desarrolladas

Listas enlazadas

Las listas enlazadas, se utilizan para manejar a las hileras de enemigos y las balas. Se implementaron dos tipos de listas enlazadas: doblemente enlazadas y circulares, según lo indicado en el enunciado del proyecto. Cabe destacar que cada uno de los métodos de las listas fueron programados, en lugar de utilizar alguna biblioteca.

Otros

Se consideró el uso de otras estructuras de datos. Se vieron opciones como pilas, colas, grafos, árboles, entre otros. No obstante, se tomó la decisión de que es preferible adecuar las estructuras de datos a utilizar a los contenidos del curso vistos hasta el momento, lo que se reduce a listas enlazadas.

Algoritmos desarrollados

Los algoritmos desarrollados se explican detalladamente utilizando JavaDoc así como documentación interna y se encuentran a disposición del lector y de manera pública en el repositorio de Github disponible en la bibliografía.

Problemas encontrados

Uso de Git y del lenguaje Java

Dado que en clases se aprende lo básico de Java y no se da una introducción al manejador de versiones Git, el primer problema encontrado se basó en el desconocimiento de estas herramientas.

Afortunadamente, la elaboración de este proyecto coincidió con una asistencia en el laboratorio de GoTouch, donde se aprendió acerca del uso de Git mediante talleres. Por otra parte, para el uso de Java fue necesario evacuar dudas constantemente respecto a la sintaxis. Principalmente se utilizaron los sitios web YouTube y

StackOverflow. Cabe destacar que el conocimiento del inglés jugó un papel fundamental para la utilización de ambas fuentes.

Desconocimiento de alguna interfaz gráfica en Java

El no conocer una interfaz gráfica para Java, requirió tiempo de investigación acerca de las opciones. Dentro de las buscadas, se encontró Java AWT y Swing. Se decidió usarlas debido su flexibilidad para crear ventanas, cargar imágenes, crear botones, manejar métodos de entrada, y su parecido a las interfaces gráficas utilizadas en el curso de Taller de Programación en Python.

Cabe destacar que, a diferencia de los cursos anteriores, esta no fue la parte que tomó más tiempo, gracias a que ya se habían desarrollado juegos anteriormente, por lo que se maneja el paradigma de diseño. Para la solución de este problema se acudió a la documentación de las librerías y a material confiable en YouTube.

Inexperiencia en programación orientada a objetos

Durante el desarrollo del juego, se continuaba aprendiendo acerca de POO. Esto, por supuesto, influyó en el tiempo de programación. Afortunadamente, se cuenta con temas como patrones de diseño, que otorgaron una solución apropiada para problemas encontrados. Por ejemplo, el uso de Factory para el desarrollo de enemigos o Adapter para el envío de información del celular a la computadora y viceversa.

Desarrollo del socket y creación de la aplicación de Android

Este, sin duda, fue el mayor problema encontrado. No solo por el desconocimiento en servidores y desarrollo de aplicaciones, sino por la poca claridad de cómo hacer funcionar el socket. A pesar de encontrar buenas guías en páginas de internet y YouTube, cuando se estableció la conexión entre los dispositivos, no se supo cuál es la mejor manera de enviar información. La solución brindada consiste en enviar Strings los cuales posteriormente son interpretados por cada dispositivo, sin embargo no se encontró algún criterio técnico que respalde que esa sea la mejor manera de enviar e interpretar la información.

Otro gran problema de esta parte, es el desconocimiento de desarrollo de aplicaciones para el sistema operativo Android. Esto requirió acudir reiteradas veces a páginas de internet (principalmente StackOverflow) para solucionar varios errores. Inicialmente no se sabía cómo cambiar los TextViews desde una clase distinta al MainActivity. Hubo errores al tratar de desarrollar hilos. Incluso todavía, al momento de finalización del proyecto, eventualmente hay errores desconocidos. La solución que debió aplicarse para esto es consultar con el profesor.

Conclusiones

Durante el desarrollo del proyecto se tuvieron importantes aprendizajes. Primeramente, se observó que la Programación Orientada a Objetos tiene múltiples beneficios y aplicaciones. Tanto las balas, enemigos, hileras, jugadores, y muchos otros aspectos en un juego se pueden representar como objetos y facilitar la programación.

Asimismo, se comprendió la utilidad de los patrones de diseño, ya que simplificaron la programación de hileras de enemigos y transmisión de datos de la computadora al celular. Lo mismo con las listas enlazadas, que son una estructura de datos que se ha descubierto que tienen una gran utilidad.

Por otra parte, quedó clara la utilidad de un manejador de versiones de código. En reiteradas ocasiones, después de programar, se llegó a la conclusión de que la versión anterior era mejor a la creada, o incluso, el programa comenzó a dar más errores o funcionar peor. En dichos casos, el uso del manejador de versiones permitió recuperar las versiones anteriores, sin necesidad de cambiar el código paso a paso.

Se concluye que uno de las principales limitaciones en la elaboración de un proyecto programado es el desconocimiento del lenguaje. Muchas veces, se descubrieron funcionalidades o bibliotecas que hubieran sido de gran ayuda al principio de la elaboración del proyecto. Además, se comenzó a comprender el funcionamiento de los servidores mediante el desarrollo del socket. Incluso se adquirió bastante conocimiento acerca de desarrollo de aplicaciones orientado a Android.

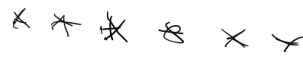

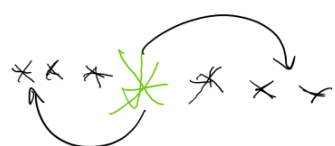

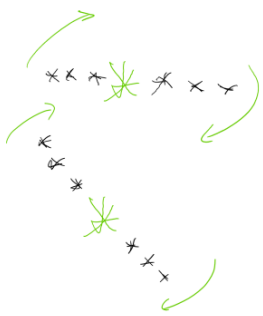
A pesar de esas limitaciones, se concluye que fue posible diseñar e implementar un juego al estilo de Space Invaders mediante listas enlazadas, utilizando patrones de diseño y Programación Orientada a Objetos.

Referencias bibliográficas

- Donalddcarling. (2016). *Blast Harrier Laser 2*. Recuperado de:
<https://donalddcarling.wordpress.com/2016/03/10/star-strike-game-build/blast-harrier-laser-2/>
- Edmobe. (2018). *Spacegame* [Repositorio de Github]. Recuperado de:
<https://github.com/edmobe/spacegame>
- Lemat Works. (s.f.). *Twinkle Night*. Recuperado de:
<https://lematworks.tumblr.com/post/155443373277/produced-by-lemat-works-twinkle-night1-2-3-4-5-6>
- Programming Experts. (2017). *Send & Receive data from Android to PC (TCP Sockets) in Android Studio PART 1*. Recuperado de:
<https://www.youtube.com/watch?v=29y4X65ZUwE>
- Programming Experts. (2017). *Send data from PC to Android using TCP Sockets Client Server in Android Studio PART 2*. Recuperado de:
<https://www.youtube.com/watch?v=LWFSGs4CG6I>
- RealTutsGML. (2014). *Beginner Java Game Development [sic]*. Recuperado de:
https://www.youtube.com/playlist?list=PLWms45O3n--4t1cUhKrqqOLeHE_sRtr0S
- RealTutsGML. (2014). *Java Game Development Series*. Recuperado de:
<https://www.youtube.com/playlist?list=PLWms45O3n--6KCNAEETGiVTEFvnqA7qCi>
- Singh, R. (s.f.). *Nave Espacial*. Recuperado de:
https://pngtree.com/freepng/space-ship_3370141.html
- York, A. (2015). *Android Studio - Tutorial 6 - Accelerometer*. Recuperado de:
<https://www.youtube.com/watch?v=Yrl2pCZC8cc&t=3s>

Anexos

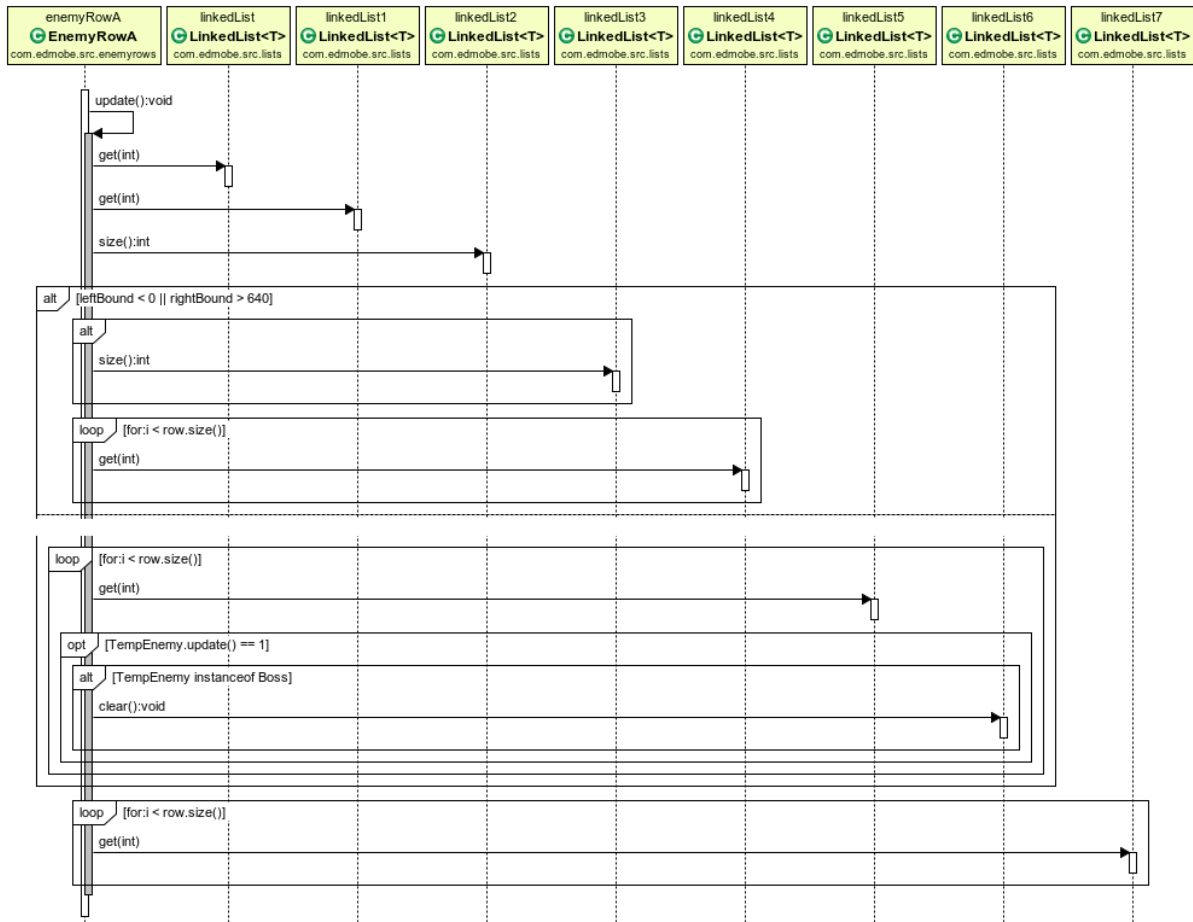
Anexo 1: tipos de filas de enemigos

Basic	Todos los enemigos de la hilera son iguales.	
Clase A	Uno de los enemigos de la hilera es jefe. Aleatoriamente se elige a un jefe. Cuando el jefe se destruye, el resto de naves desaparece. Para destruir el jefe, se requieren entre 2 a 5 disparos	
Clase B	Clase A con una variación. El jefe se intercambia rápidamente con el resto de los miembros de la hilera. Es una lista doblemente enlazada.	
Clase C	Igual que la A pero si el jefe se destruye, otro toma su lugar aleatoriamente. Es una lista circular.	
Clase D	Igual que la C pero los miembros de la hilera tienen diferentes resistencias. La Hilera se mantiene ordenada de mayor a menor resistencia, utilizando bubble sort.	
Clase E	Igual que la C con algunas variaciones. La hilera no es horizontal, constantemente rota según las manecillas del reloj. El jefe siempre está en el centro. Es una lista doble y circular.	

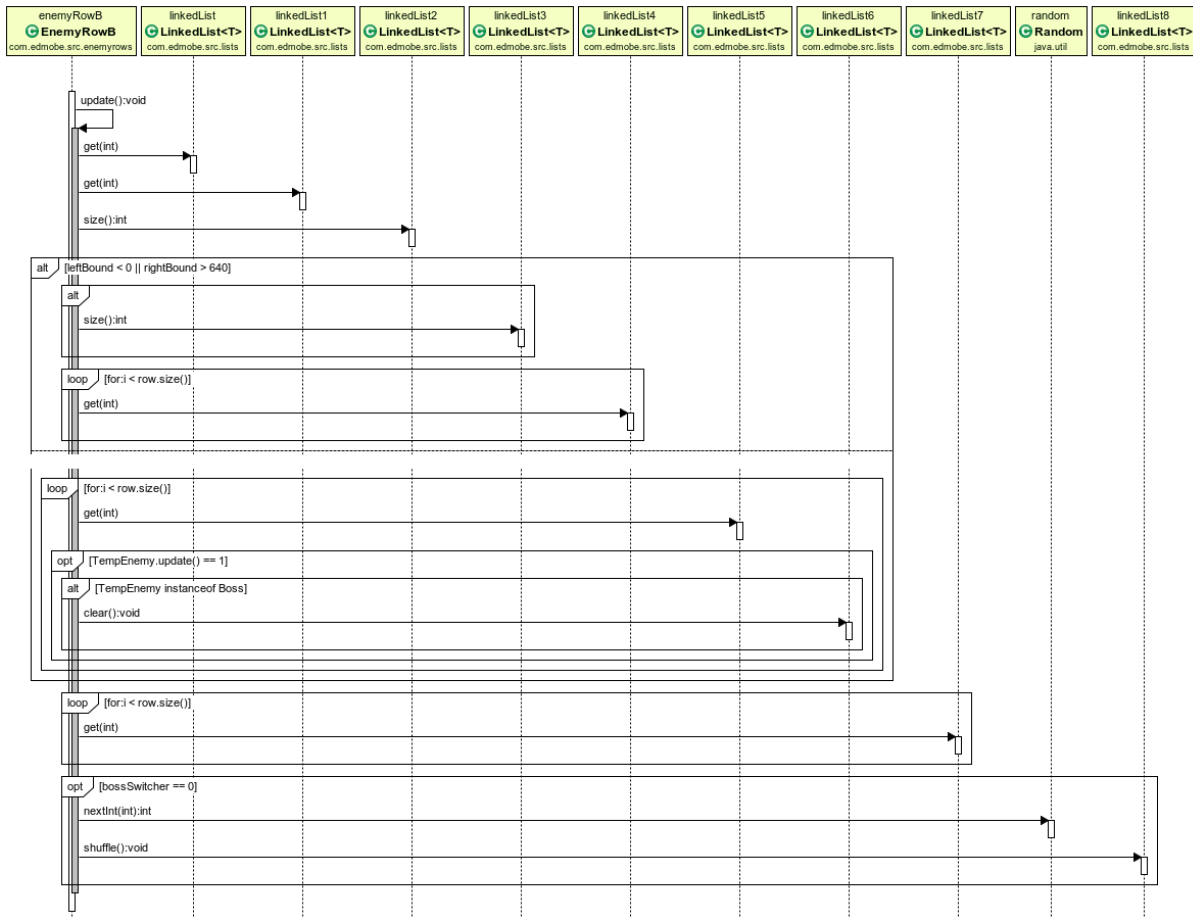
Fuente: Enunciado del proyecto

Anexo 2: diagramas de secuencia para las filas de enemigos clase A, B, C y D

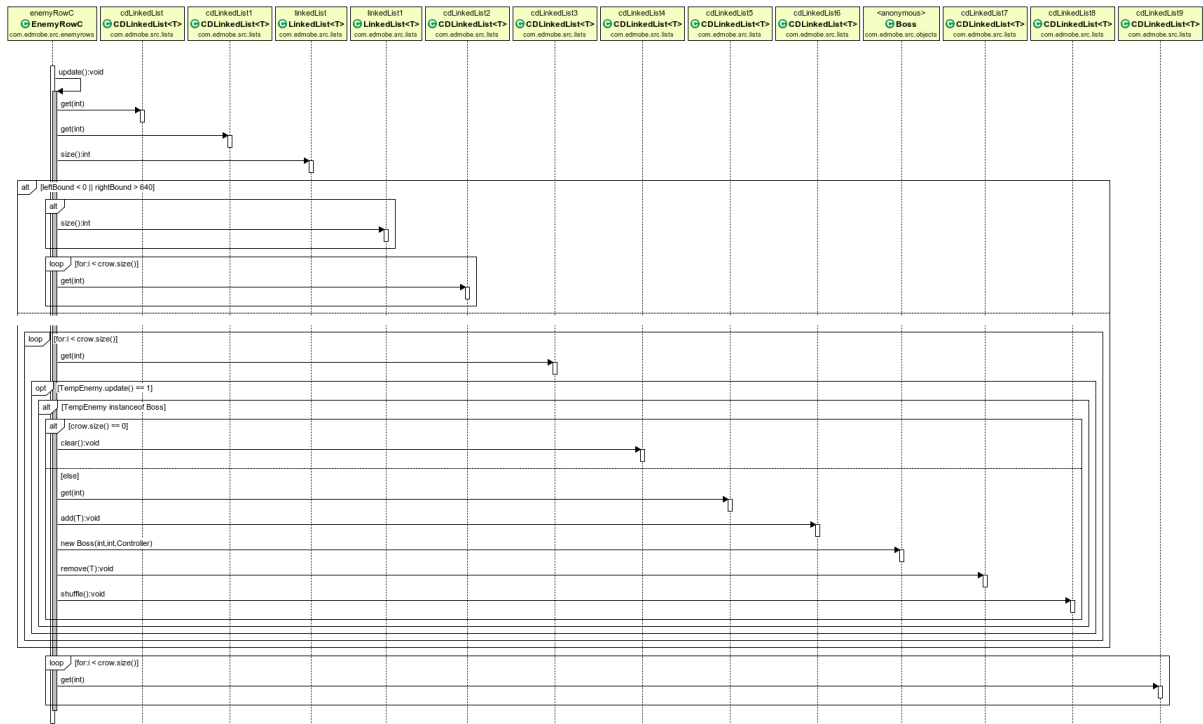
Clase A



Clase B



Clase C



Clase D

