



TALLER DE PROGRAMACIÓN I  
(75.42) CURSO VEIGA

# Informe técnico - Duck Game

2do Cuatrimestre 2024

Ascencio Felipe Santino	Zielonka Axel
110675	110310

## 1. Introducción

Este informe técnico describe el funcionamiento del programa a nivel de código. Su objetivo es proporcionar a futuros desarrolladores una comprensión profunda del sistema para que puedan mejorar el proyecto o adaptarlo según sus necesidades. Asimismo, está pensado para aquellos interesados en conocer el funcionamiento interno del juego.

## 2. Organización de los Directorios

A continuación se presenta la estructura de directorios y una breve descripción de cada uno.

### 2.1. Directorio `documentation`

Este directorio contiene todos los manuales e informes necesarios para la documentación y uso del juego.

### 2.2. Directorio `SDL`

Contiene las dependencias para las bibliotecas *SDL2*, necesarias para el manejo gráfico y de sonido en el juego.

### 2.3. Directorio `src`

En este directorio se almacena todo el código fuente del juego, dividido en varias subcarpetas, descritas a continuación:

#### 2.3.1. Directorio `client`

Incluye el código relacionado con el cliente, encargado de la interfaz gráfica y la comunicación con el servidor.

#### 2.3.2. Directorio `server`

Contiene el código del servidor, que gestiona la lógica del juego y la comunicación con los clientes.

#### 2.3.3. Directorio `common`

Guarda el código compartido entre los directorios `client` y `server`. Aquí se encuentran archivos con métodos y estructuras utilizadas en ambos programas.

#### 2.3.4. Directorio `tests`

Contiene el código de los 'tests' que verifican el correcto funcionamiento del 'Protocolo' y la 'Comunicación' Cliente-Servidor.

#### 2.3.5. Directorio `editor`

Actualmente solo se encuentra un esqueleto de la estructura para un futuro 'editor', la idea sería en próximas actualizaciones añadirlo para agrandar las funcionalidades del juego.

### 2.3.6. Directorio data

Este directorio contiene todos los recursos multimedia del juego, organizados en las siguientes categorías:

- Sprites.
- Música.
- Sonidos.
- Pantallas de victoria y derrota.
- Fuentes de texto.
- Mapas.
- Archivos de inicialización de la matriz de mapas (utilizados por el servidor).

### 2.3.7. Directorio SDL2pp

Este directorio contiene todo lo necesario para utilizar la librería "SDL2pp" para el apartado gráfico y de sonido del juego.

### 2.3.8. Directorios cmake y cmake-build-debug

Contienen la configuración y ejecución de los archivos para la compilación del proyecto mediante CMake.

### 2.3.9. Archivo de configuración config-juego.yaml

Este archivo permite la 'configuración-modificación' de los aspectos personalizables del juego.

## 2.4. Archivos adicionales en el directorio raíz

En el directorio raíz se encuentran los archivos principales de instalación y ejecución del juego, incluyendo el archivo **README**, el cual provee instrucciones básicas para la instalación y uso del juego, facilitando el proceso a usuarios sin experiencia previa.

## 3. Documentación del Código

### 3.1. Servidor

El servidor es responsable de:

- Controlar la lógica del juego.
- Comunicar el estado del juego a todos los jugadores conectados.
- Administrar la sincronización y la coexistencia de los jugadores, eliminándolos si están inactivos o cuando termina la partida.

A continuación se adjuntarán diagramas de clases y de secuencia para las partes más relevantes (UML).

### 3.1.1. Diagrama de clase: Aceptador

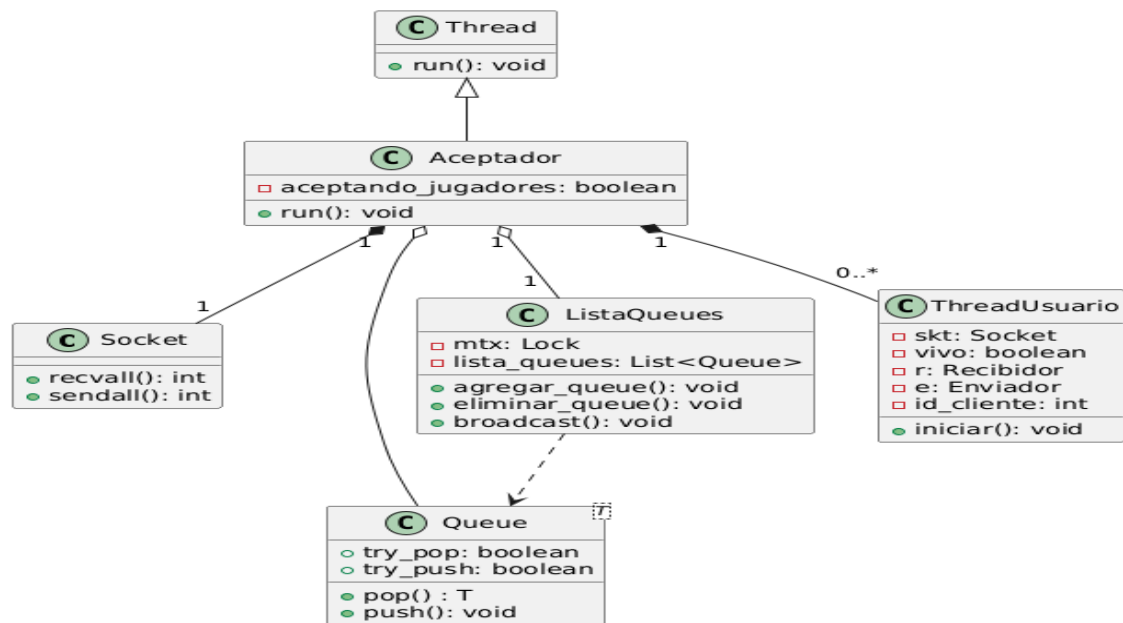


Figura 1: Diagrama de Clases UML : 1.

### 3.1.2. Diagrama de clase: Arma

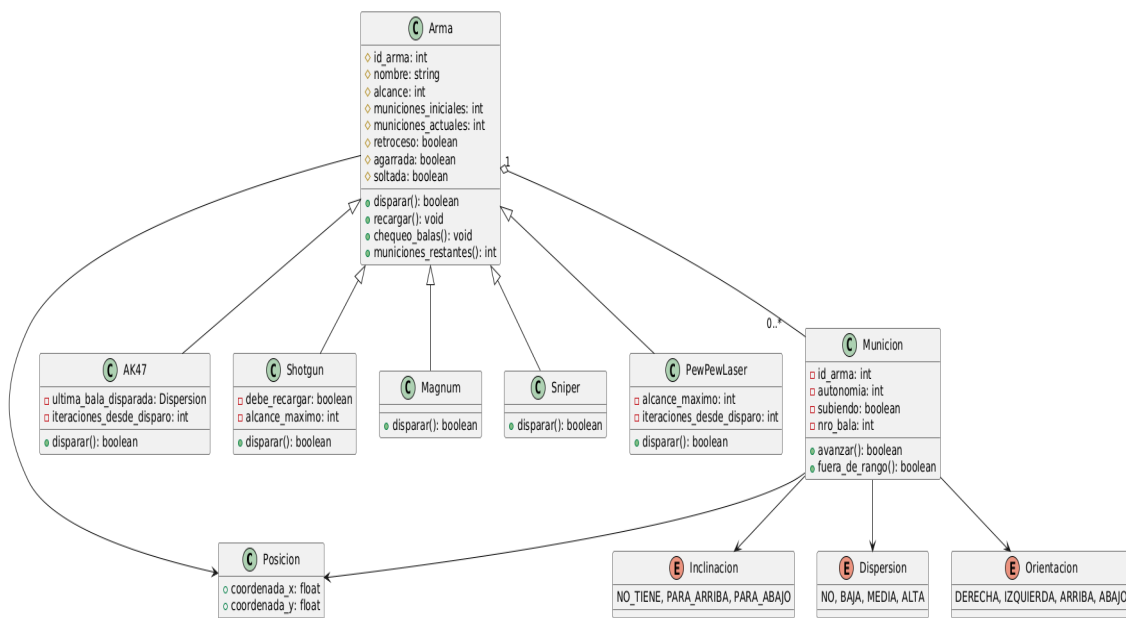


Figura 2: Diagrama de Clases UML : 2.

### 3.1.3. Diagrama de clase: Estado del Juego

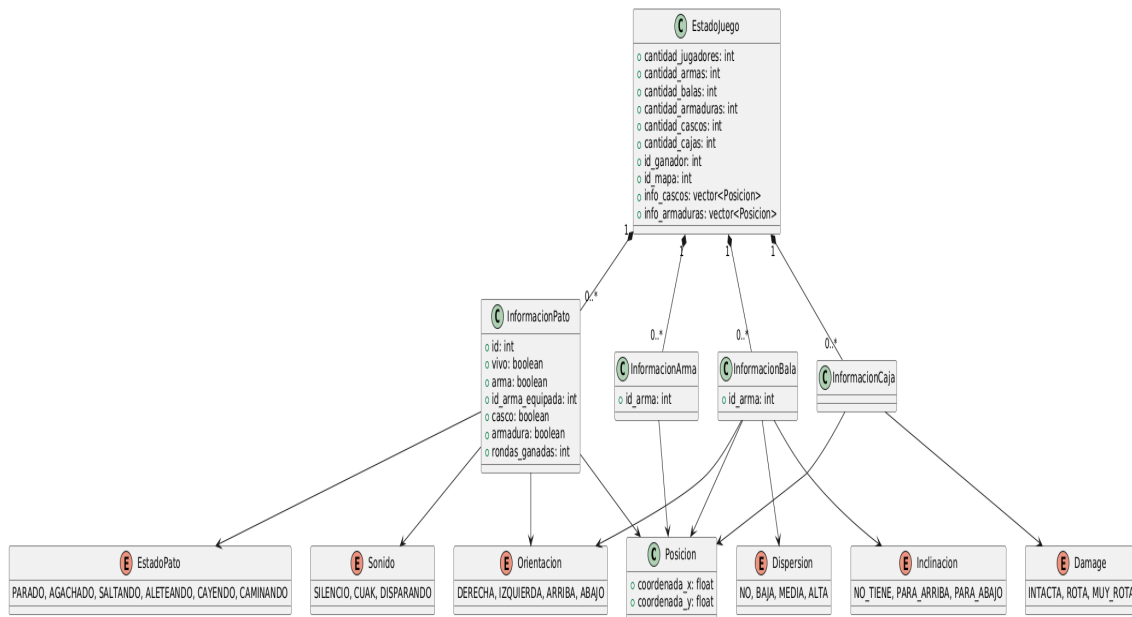


Figura 3: Diagrama de Clases UML : 3.

### 3.1.4. Diagrama de clase: Gameloop

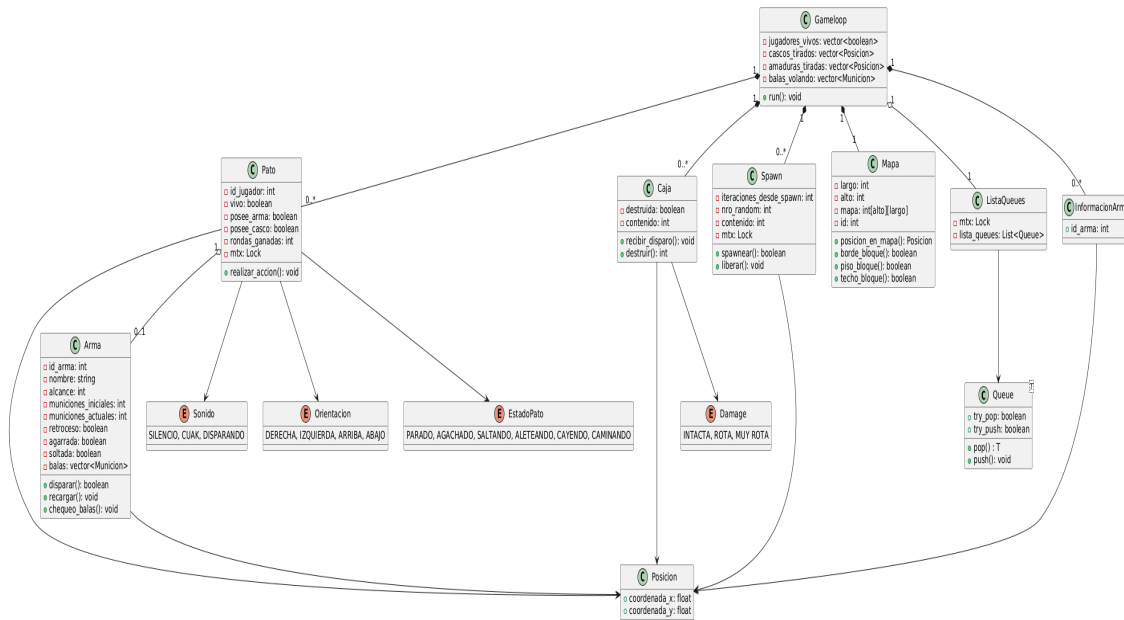


Figura 4: Diagrama de Clases UML : 4.

### 3.1.5. Diagrama de clase: Server

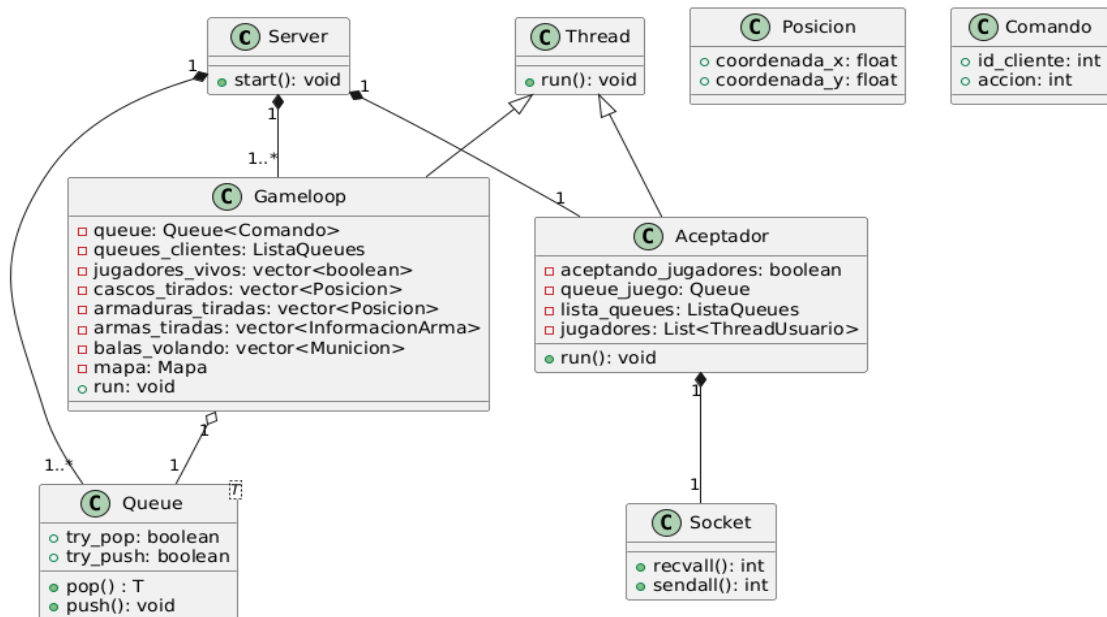


Figura 5: Diagrama de Clases UML : 5.

### 3.1.6. Diagrama de clase: Threads del Cliente

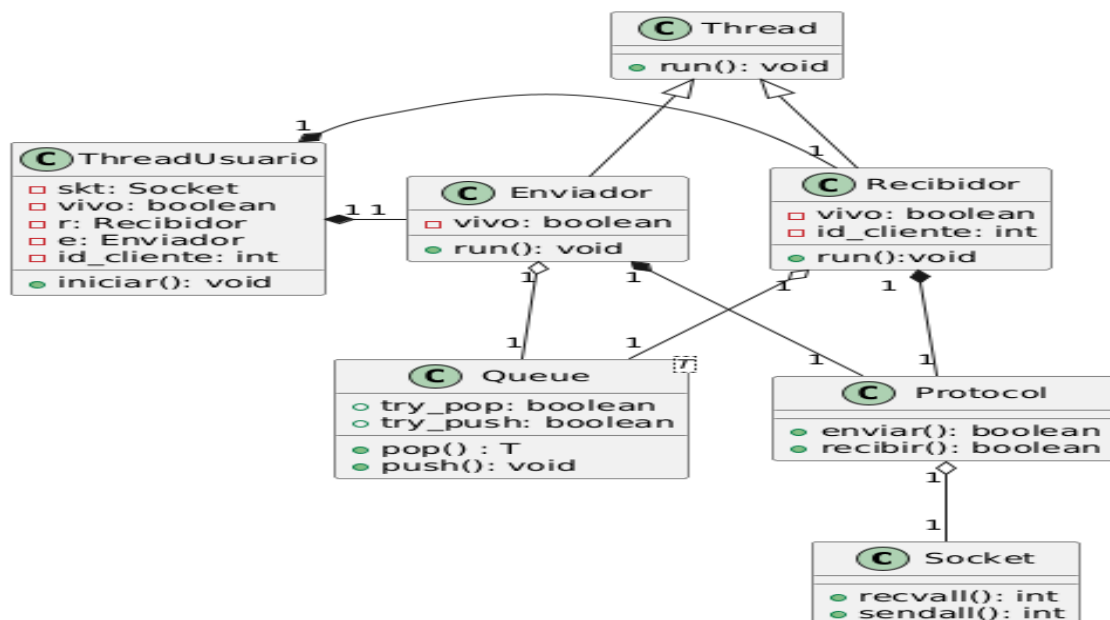


Figura 6: Diagrama de Clases UML : 6.

### 3.1.7. Diagrama de secuencia: Aceptador

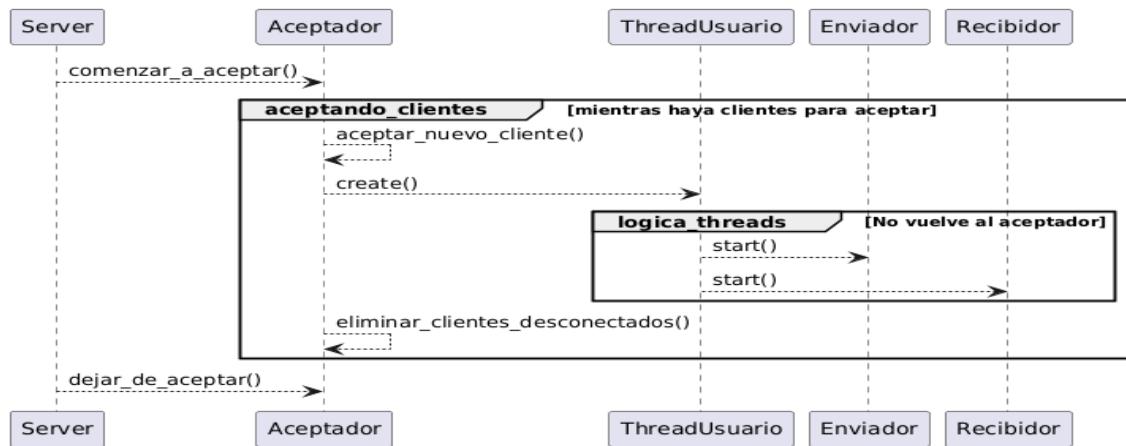


Figura 7: Diagrama de Secuencia: 1.

### 3.1.8. Diagrama de secuencia: Hilo Enviador

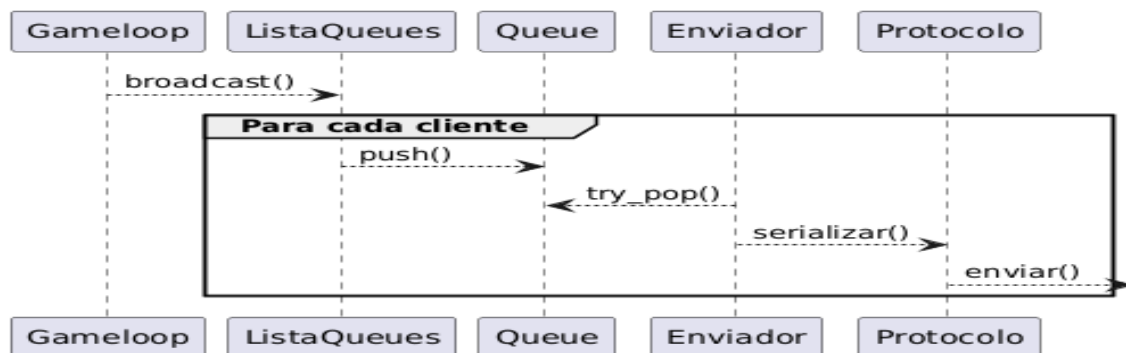


Figura 8: Diagrama de Secuencia: 2.

### 3.1.9. Diagrama de secuencia: Hilo Recibidor

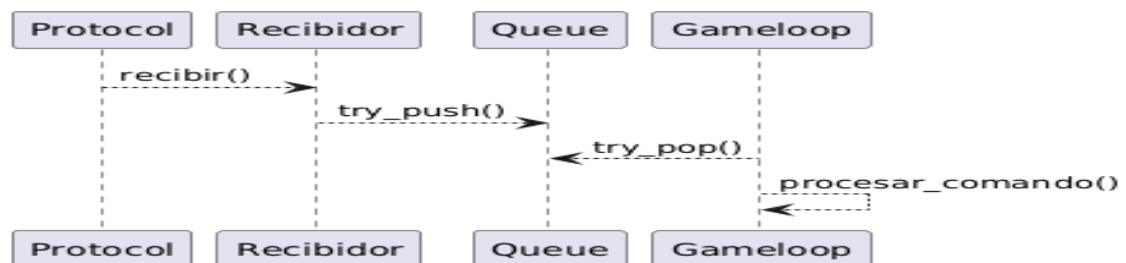


Figura 9: Diagrama de Secuencia: 3.

### 3.1.10. Diagrama de secuencia: Gameloop

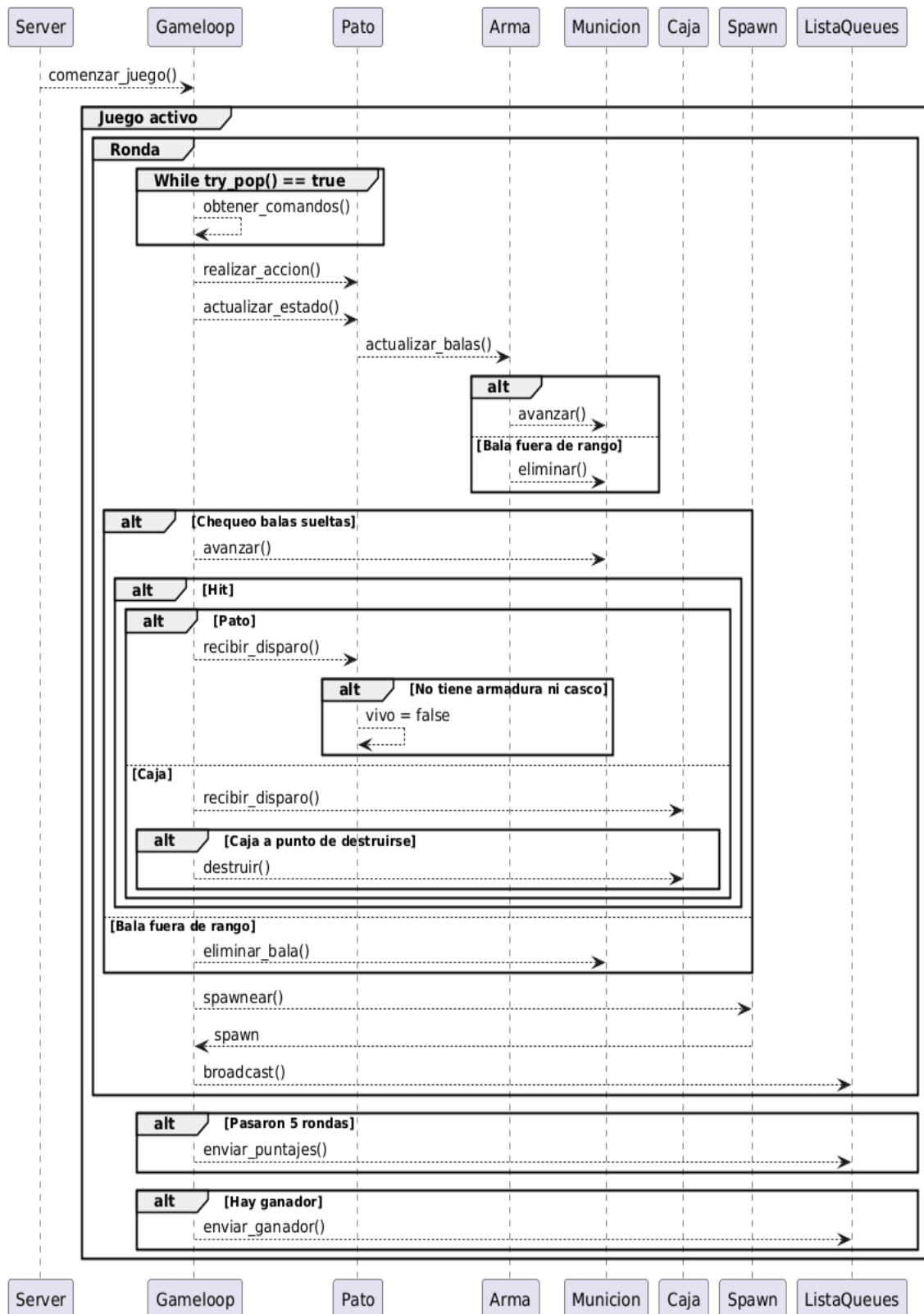


Figura 10: Diagrama de Secuencia: 4.



### 3.2. Cliente

El cliente es el encargado de:

- Representar visualmente el estado del juego para el jugador.
- Enviar las actualizaciones del jugador al servidor.
- Gestionar y utilizar las texturas y sonidos requeridos.

A continuación se anexarán diagramas de clases y de secuencia para las partes más relevantes (UML).

#### 3.2.1. Diagrama de clase: Comunicación con el Server

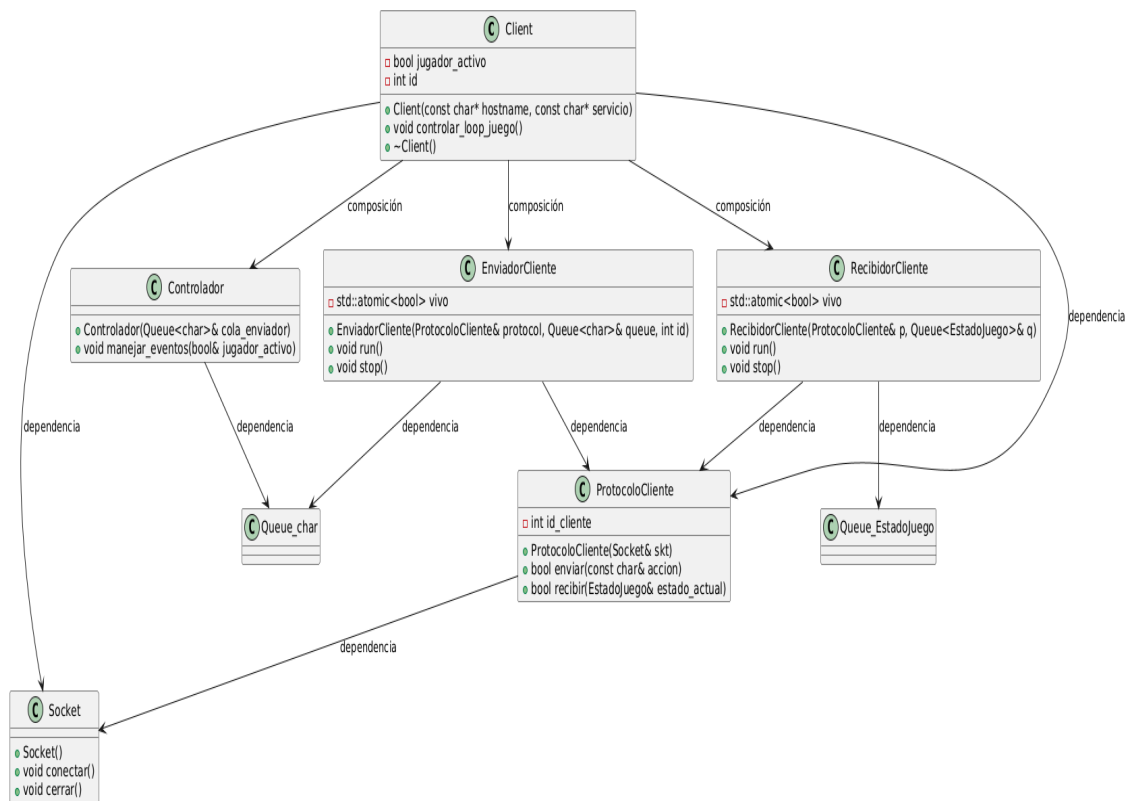


Figura 11: Diagrama de Clases UML : 7.

### 3.2.2. Diagrama de clase: Dibujo de la partida

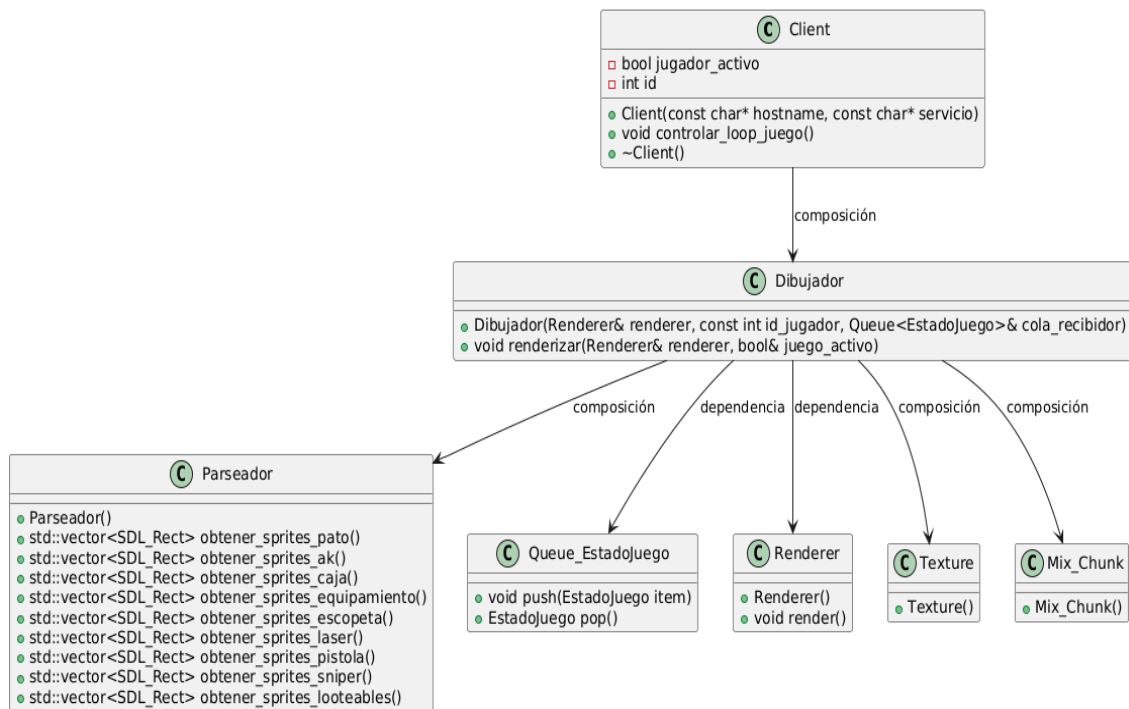


Figura 12: Diagrama de Clases UML : 8.

### 3.2.3. Diagrama de secuencia: Conexión con el Servidor

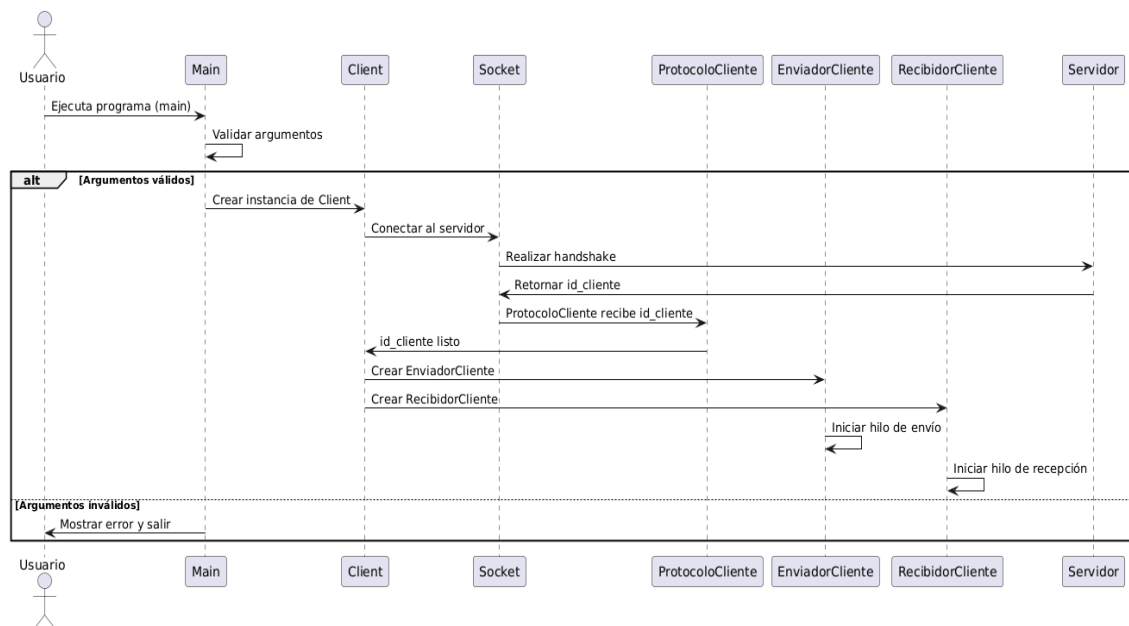


Figura 13: Diagrama de Secuencia: 5.

### 3.2.4. Diagrama de secuencia: Hilo Enviador

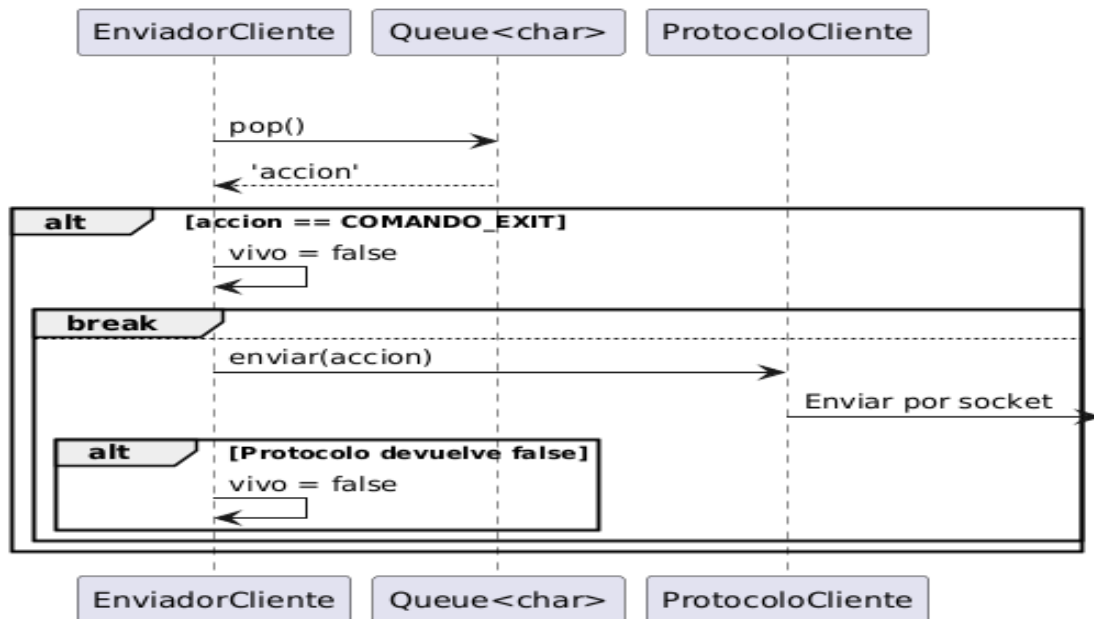


Figura 14: Diagrama de Secuencia: 6.

### 3.2.5. Diagrama de secuencia: Hilo Recibidor

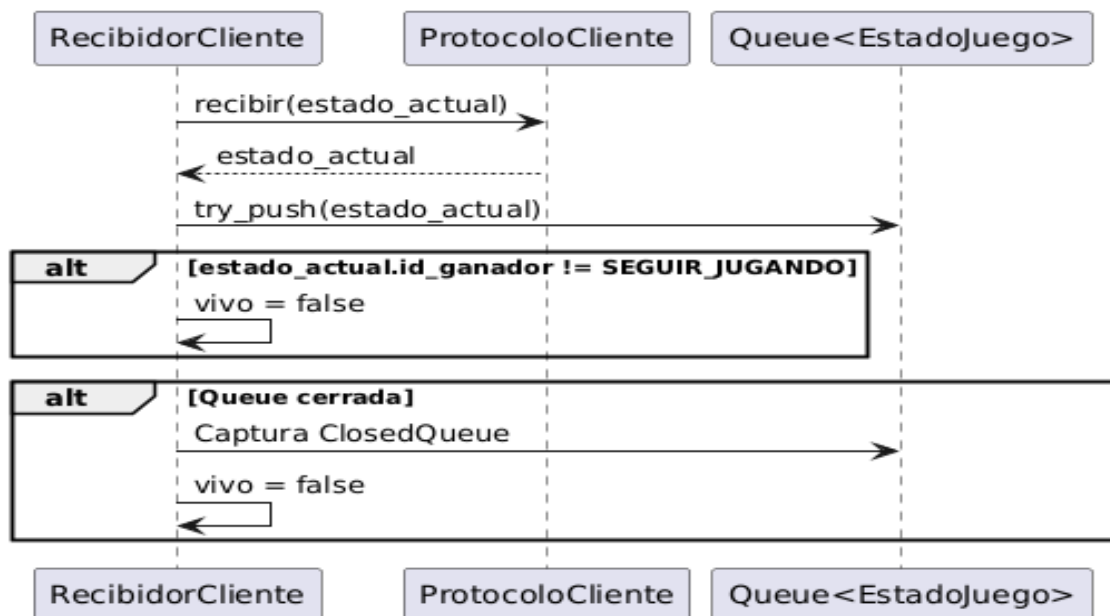


Figura 15: Diagrama de Secuencia: 7.

### 3.2.6. Diagrama de secuencia: Lógica del Cliente

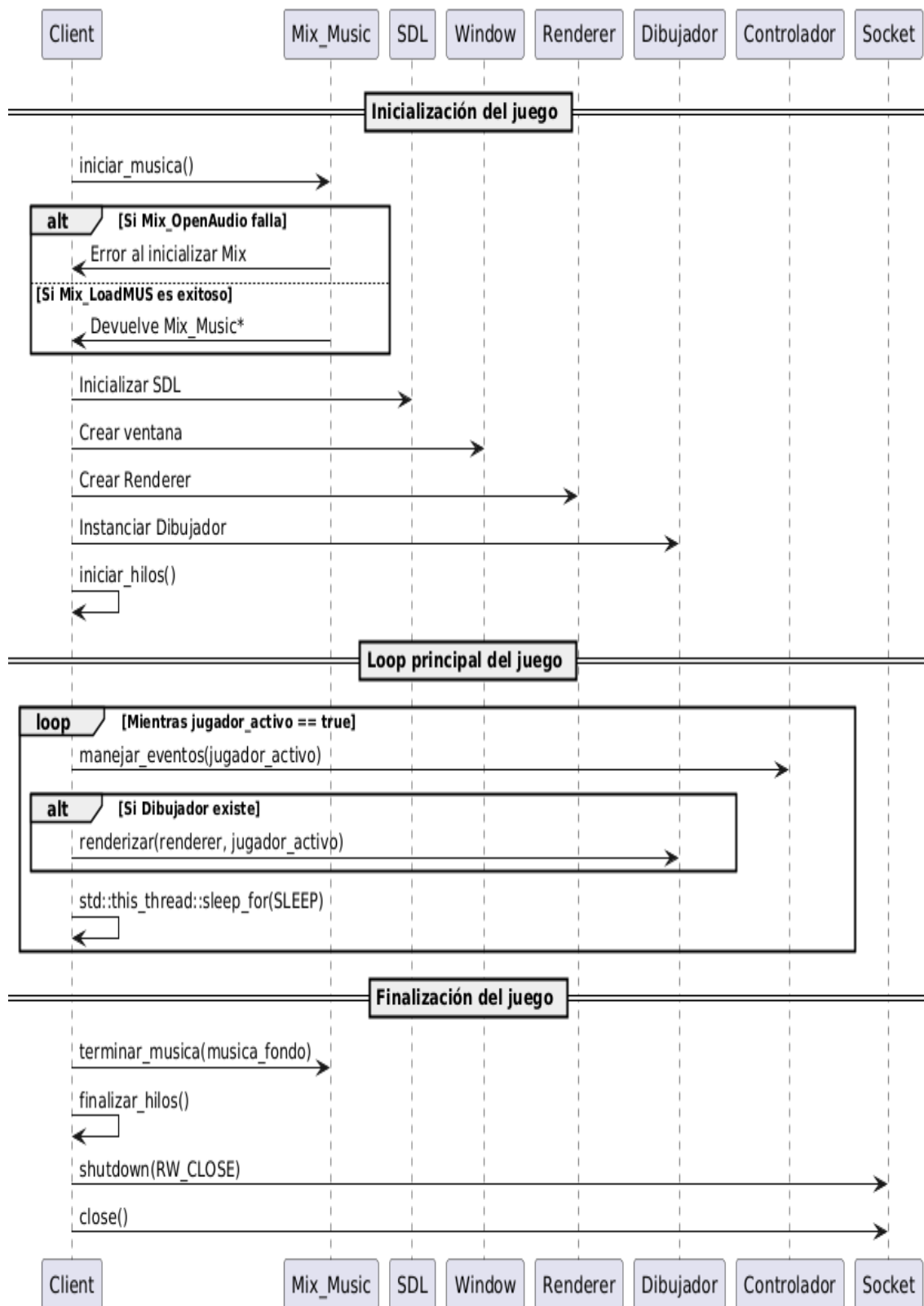


Figura 16: Diagrama de Secuencia: 8.

### 3.2.7. Diagrama de secuencia: Control de eventos del teclado

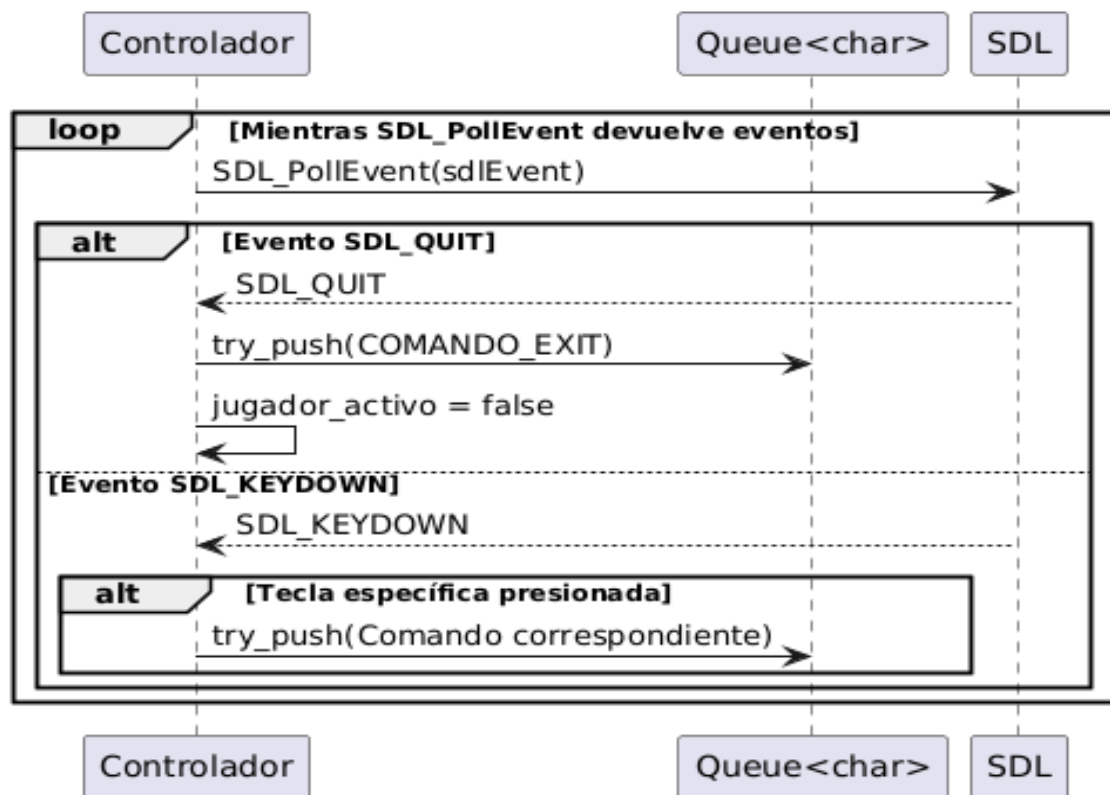


Figura 17: Diagrama de Secuencia: 9.

### 3.2.8. Diagrama de secuencia: Dibujo del estado actual del juego

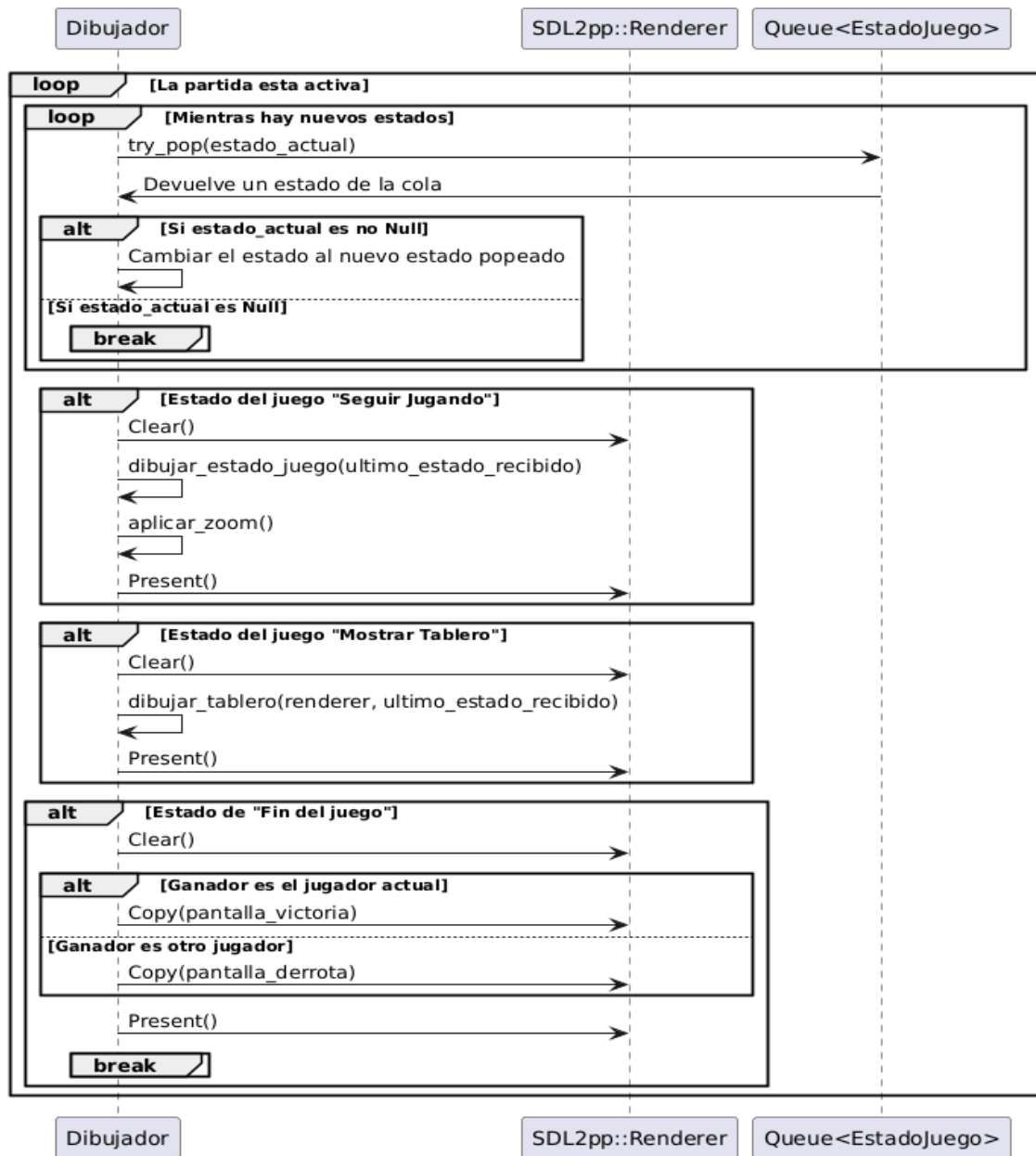


Figura 18: Diagrama de Secuencia: 10.

## 4. Conclusión

Esperamos que esta documentación facilite la comprensión del funcionamiento del juego. Invitamos a los interesados a mejorar este proyecto, implementando sus propias ideas y funcionalidades. Si decides hacerlo, no dudes en contactarnos; nos encantaría probar tus contribuciones.