

PROGRAMACIÓN ORIENTADA A OBJETOS

TRABAJO INTEGRADOR

Profesor

Esp. Ing. César Omar Aranda

Integrantes

Romero, Gonzalo

Torres López, Mía

Resumen

En el presente informe se desarrollará el trabajo final integrador de la cátedra Programación orientada a objetos de la carrera de ingeniería en Mecatrónica de la Universidad Nacional de Cuyo. El trabajo consiste en la programación de una interfaz gráfica para la implementación virtual del juego de cartas “Chancho va”. Para ello se ha planteado el desarrollo de dos modos de juego, uno para un solo jugador compitiendo contra algoritmos desarrollados y otro multijugador utilizando conexiones servidor-cliente. Para realizar esta tarea se utilizó el framework multiplataforma orientado a objetos Qt y el lenguaje de programación C++.

Introducción

En el siguiente trabajo se utiliza el paradigma de programación Orientado a Objetos para la creación de el juego de cartas “Chanco Va”. En el mismo, se reparten cuatro cartas a cada jugador, el objetivo es formar lo más rápido posible la combinación de 4 cartas del mismo número y cantar “Chanco!”, apoyando la mano en el centro de la mesa. El resto de los jugadores también deben colocar la mano sobre la mesa y el último jugador en apoyar la mano aumenta su puntaje. Una vez que el jugador llega a un puntaje igual a siete (es decir, la cantidad de caracteres de la palabra “CHANCHO”) se da por finalizado el juego, declarando como perdedor a dicho jugador.

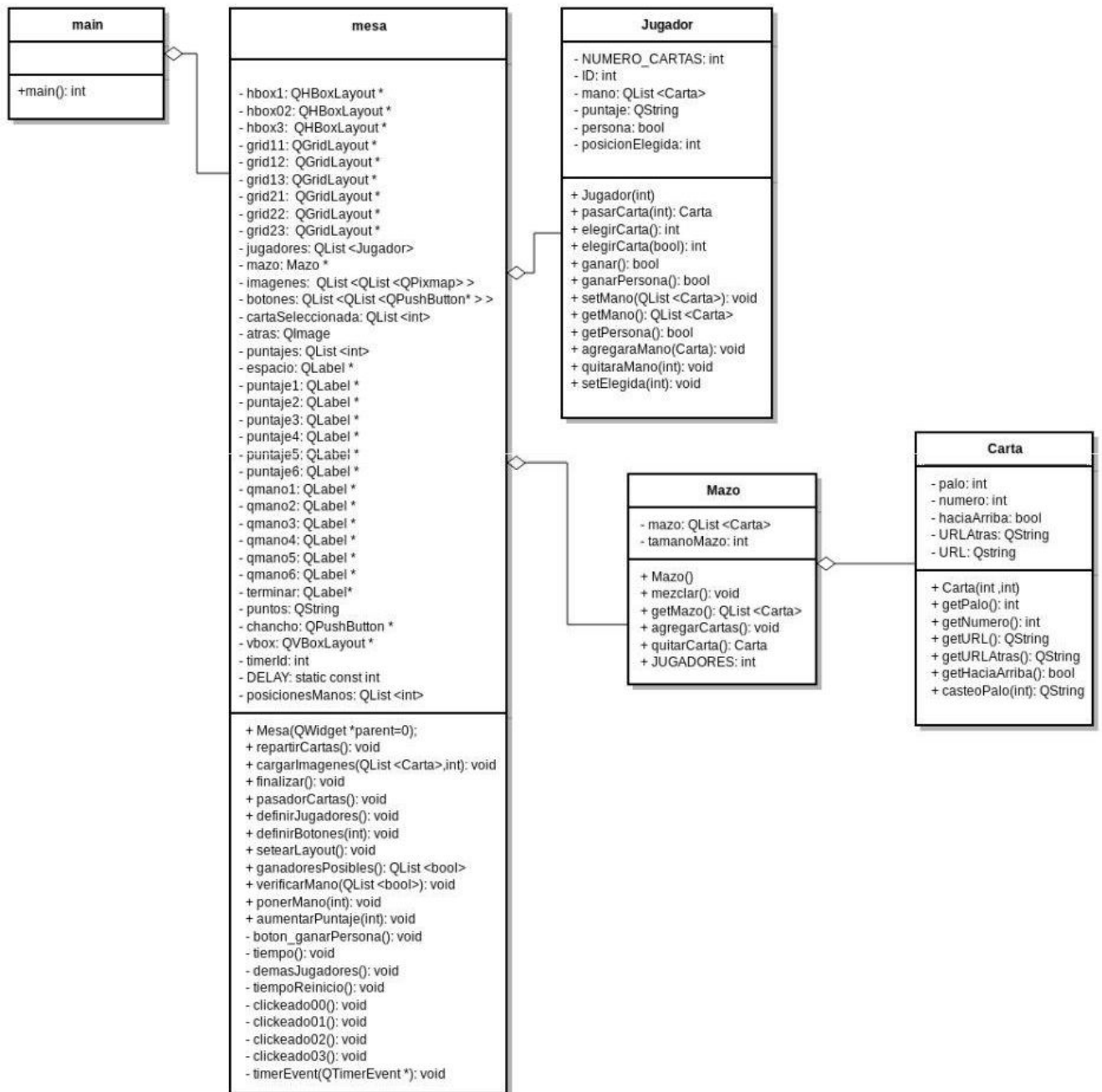
Para el desarrollo del trabajo se utilizó el editor de texto Atom y el framework Qt con lenguaje de programación C++.

En el informe se comenzará explicando la etapa de diseño aplicando diagramas de clases en UML, realizando una breve explicación de las clases que lo componen. Luego se detallará el funcionamiento de la aplicación. Por último se expondrán las conclusiones obtenidas.

Un jugador

Etaapa de diseño

El diagrama de clases de modo de un jugador se mostrará a continuación:



A continuación, se describen las clases utilizadas:

Clase Carta: Esta clase es la que se encarga de representar los objetos carta que tendrán los jugadores, detallando su número y palo y si la carta está dada vuelta o no. Además se almacenará el path a la imagen que representa a dicha carta.

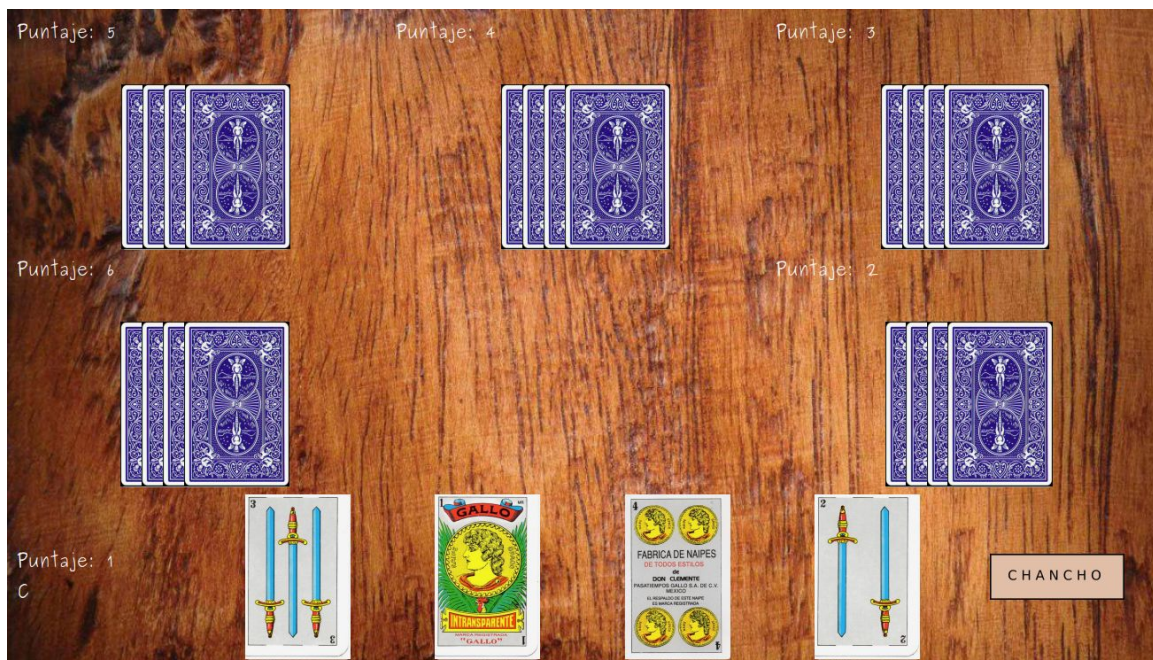
Clase Mazo: Esta clase contendrá una lista de objetos carta para formar un mazo de acuerdo a la cantidad de jugadores. Esta clase contiene métodos para mezclar el mazo para asegurar la aleatoriedad a la hora de repartir las cartas a los jugadores, también para agregar y quitar cartas del mazo.

Clase Jugador: Esta clase genera la lista de objetos carta que tendrá cada jugador, para poder pasar las mismas a otros jugadores. Cada jugador puede elegir qué carta pasar o, de lo contrario, se envía una carta por default. El jugador tiene la opción de cantar “Chancho” y posteriormente, la clase Mesa comprobará si fue correcta o no la acción del jugador, verificando la mano del jugador para ver si sus cartas son iguales. Cada jugador conoce su propio puntaje y su identificación dentro del juego.

Clase Mesa: Esta clase es la que implementará la lógica del juego y armará la interfaz gráfica. Como se mencionó anteriormente, verifica las manos de los jugadores, declarará qué jugador es el que pierde la ronda e indica los tiempos de juego a través de timers. Además, realiza el manejo de eventos.

Aplicación

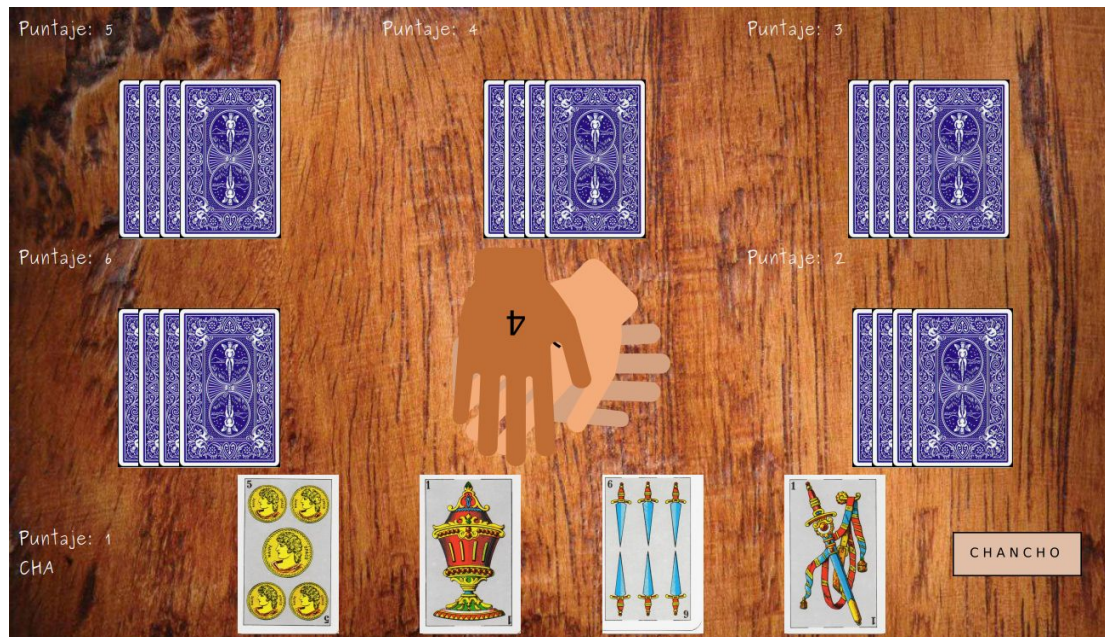
Cuando el usuario de la aplicación inicia el juego, verá la siguiente pantalla:



En la misma, el usuario puede ver su mano de cartas y los puntajes de todos los jugadores. Dispone de un botón titulado “CHANCHO” para poder poner la mano en la mesa. Las cartas son botones, de tal modo que cuando el usuario haga click sobre una de las

cartas, ésa será la carta que pasará al jugador que está a su derecha. En cada ciclo de juego (definidos por la clase Mesa) el jugador deberá pasar y recibir una carta. Si no selecciona una nueva carta en cada ciclo, se enviará la carta que esté en la posición de la última carta enviada.

En la siguiente pantalla se observa como es la acción de colocar las manos en la mesa cuando algún jugador canta “Chancho”:



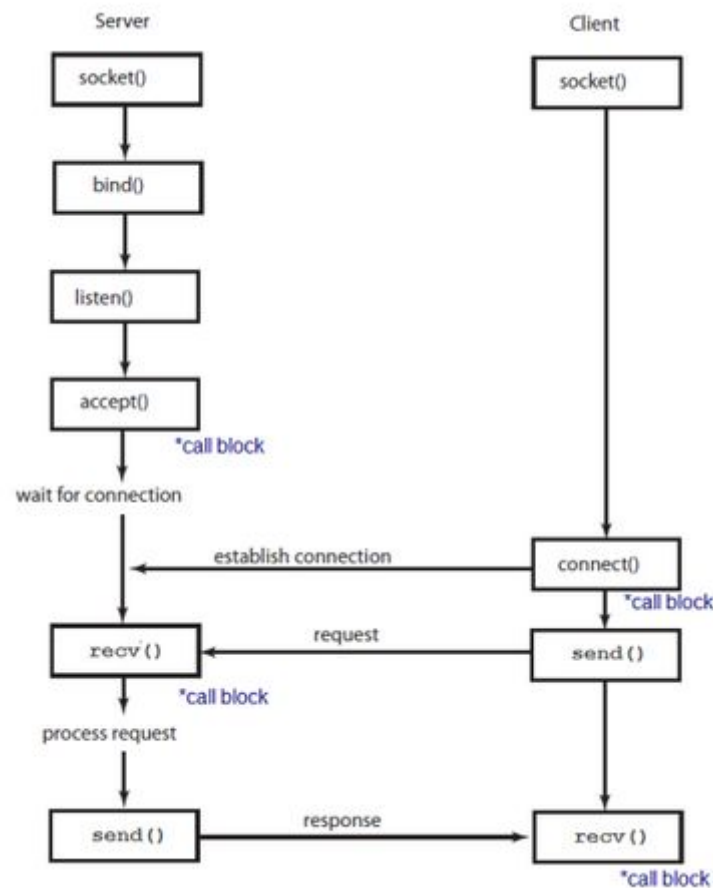
En cada ronda del juego, el último jugador en poner la mano verá como su puntaje va aumentando, sumando las letras de “CHANCHO”. En caso de que el jugador presione el botón “CHANCHO” sin que todas sus cartas sean del mismo número, su puntaje aumentará. Cuando un jugador sumó un puntaje igual a siete (es decir, que tiene la palabra CHANCHO completa), se lo declara como perdedor, como se aprecia en la siguiente pantalla:



La partida se considera entonces finalizada y la clase Mesa se encargará de reiniciar la interfaz para que el jugador pueda comenzar una nueva partida.

Multijugador

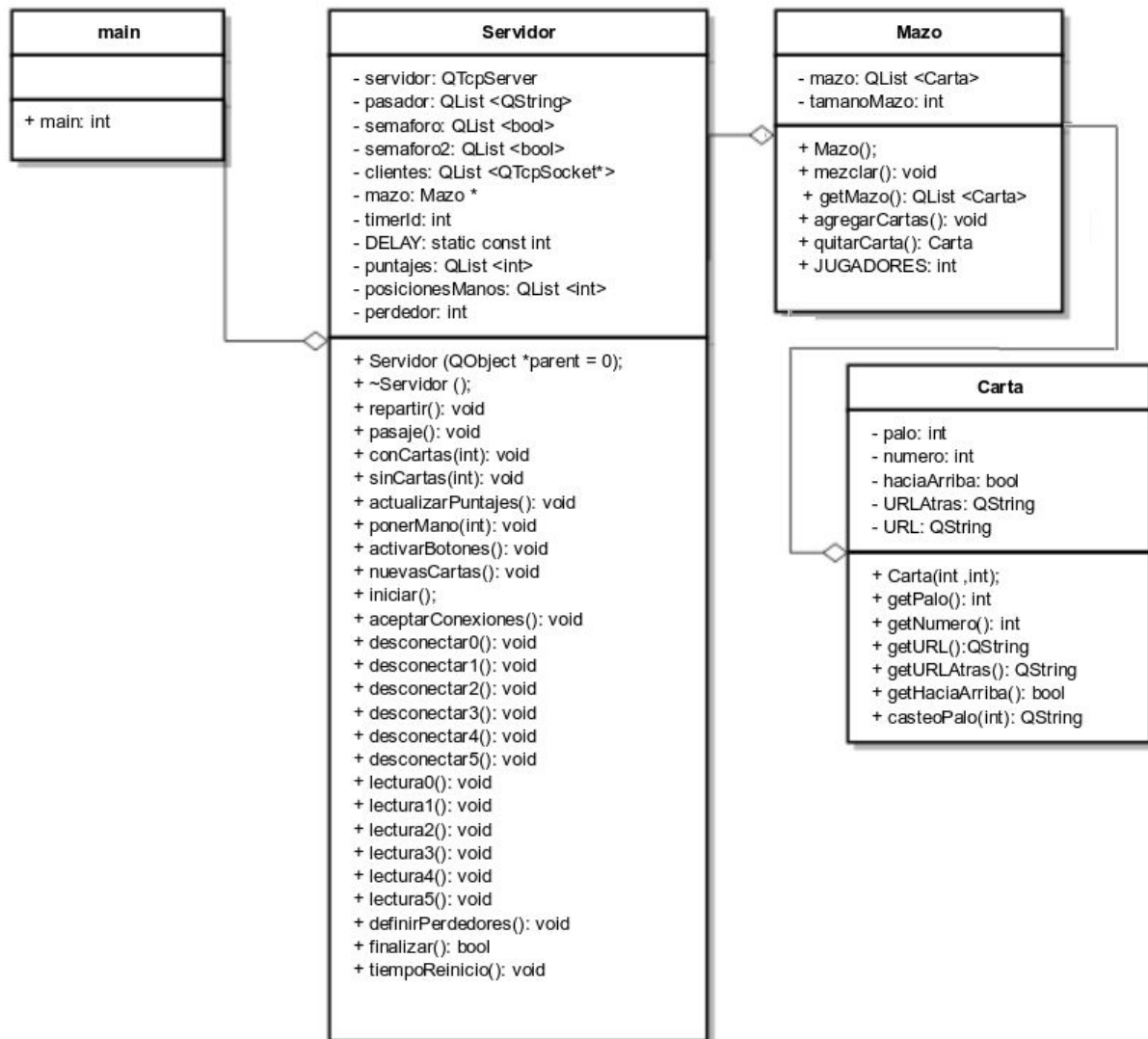
Diagrama de actividad



Primero inicializamos el servidor, el cual escuchará y esperará las conexiones de los clientes. En nuestro programa, el servidor espera por seis conexiones para enviar un mensaje a los clientes para que creen la interfaz del juego. Luego, durante toda la ejecución de la aplicación, el servidor y cliente se enviarán distintos mensajes para el correcto funcionamiento del juego.

Etapa de diseño

El diagrama de clases de modo multijugador para el servidor se mostrará a continuación:



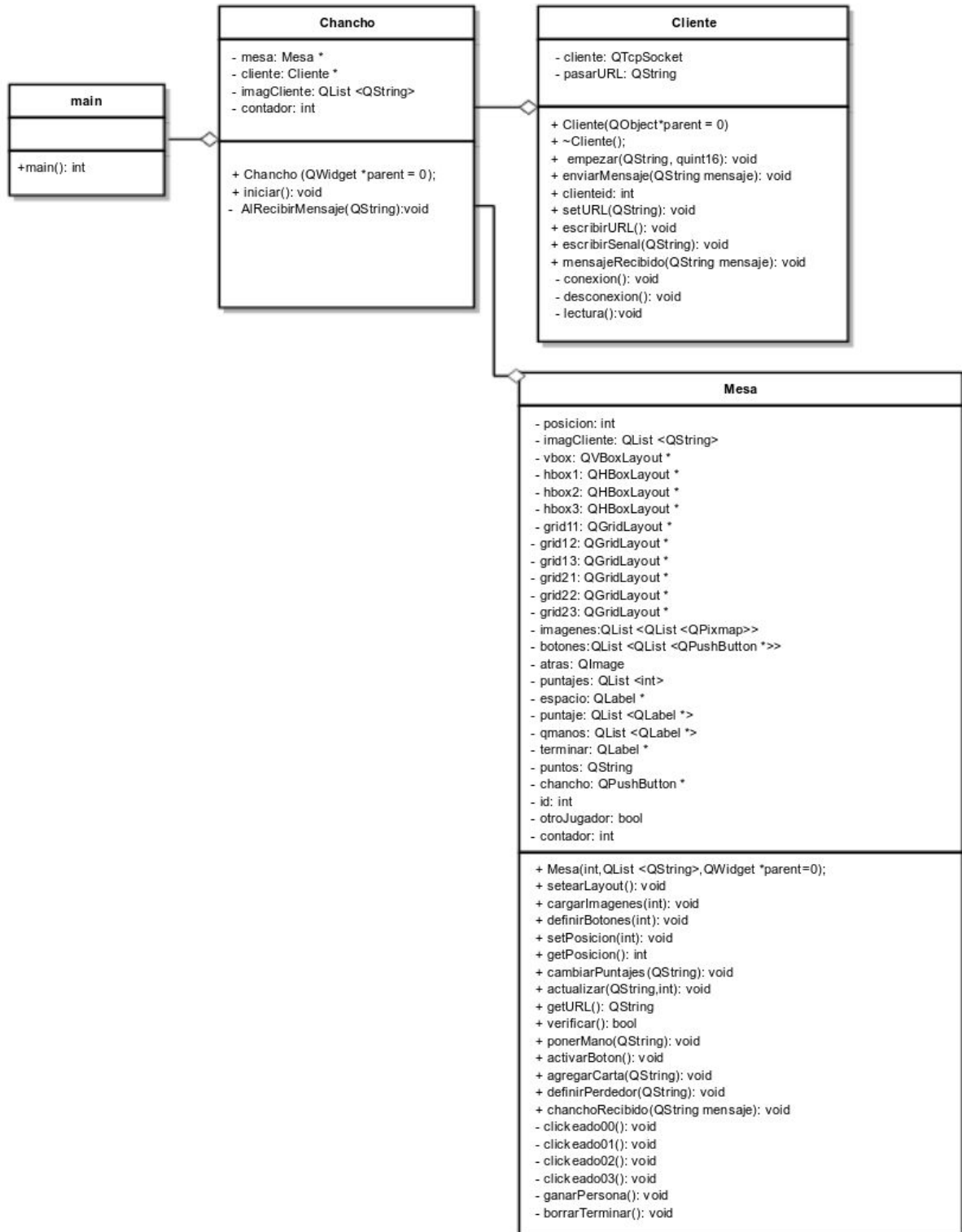
Clase Carta: Esta clase cumple las mismas funciones que la clase Carta del modo un jugador, es decir, determina el número y palo de cada carta y si la misma está dada vuelta o no.

Clase Mazo: Esta clase cumple las mismas funciones que la clase Mazo del modo un jugador, es decir, genera una lista de cartas a la que se le agregarán y quitarán cartas y que luego se mezclará.

Clase Servidor: Esta clase crea el servidor, acepta las conexiones de los clientes, lee lo que éstos le escriben, define los perdedores, genera los timers, actualiza los puntajes,

coordina el pasaje de cartas y reparte. Para poder realizar estas acciones, posee una lista de clientes y un objeto mazo.

El diagrama de clases de modo multijugador para el cliente se mostrará a continuación:



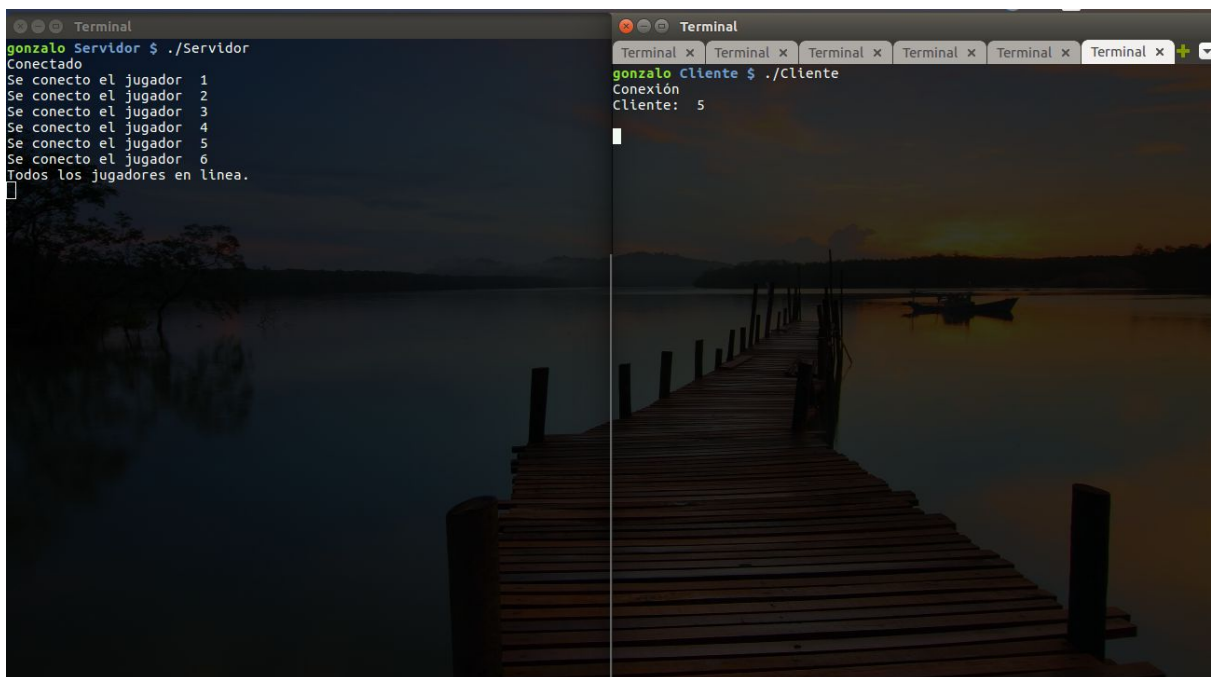
Clase Cliente: genera el socket de comunicación con el servidor, manda y recibe mensajes con el mismo. Los mensajes recibidos son derivados a la clase Chancho para su interpretación. Posee los id de los clientes para su identificación.

Clase Chancho: interpreta los mensajes provenientes del clientes y provenientes de la mesa y define las acciones correspondientes de acuerdo a la señal recibida.

Clase Mesa: Esta clase se encarga de generar la interfaz que verá cada cliente y de actualizar la misma en cada ronda del juego. Además, verifica si la mano del cliente está compuesta por cartas del mismo número.

Aplicación

El primer paso a realizar es conectar el servidor y luego conectar los clientes a él. El servidor le indicará a cada cliente cuál será el número que lo identificará durante la partida:



```
Terminal
gonzalo Servidor $ ./Servidor
Conectado
Se conecto el jugador 1
Se conecto el jugador 2
Se conecto el jugador 3
Se conecto el jugador 4
Se conecto el jugador 5
Se conecto el jugador 6
Todos los jugadores en línea.

Terminal
gonzalo Cliente $ ./Cliente
Conexión
Cliente: 5
```

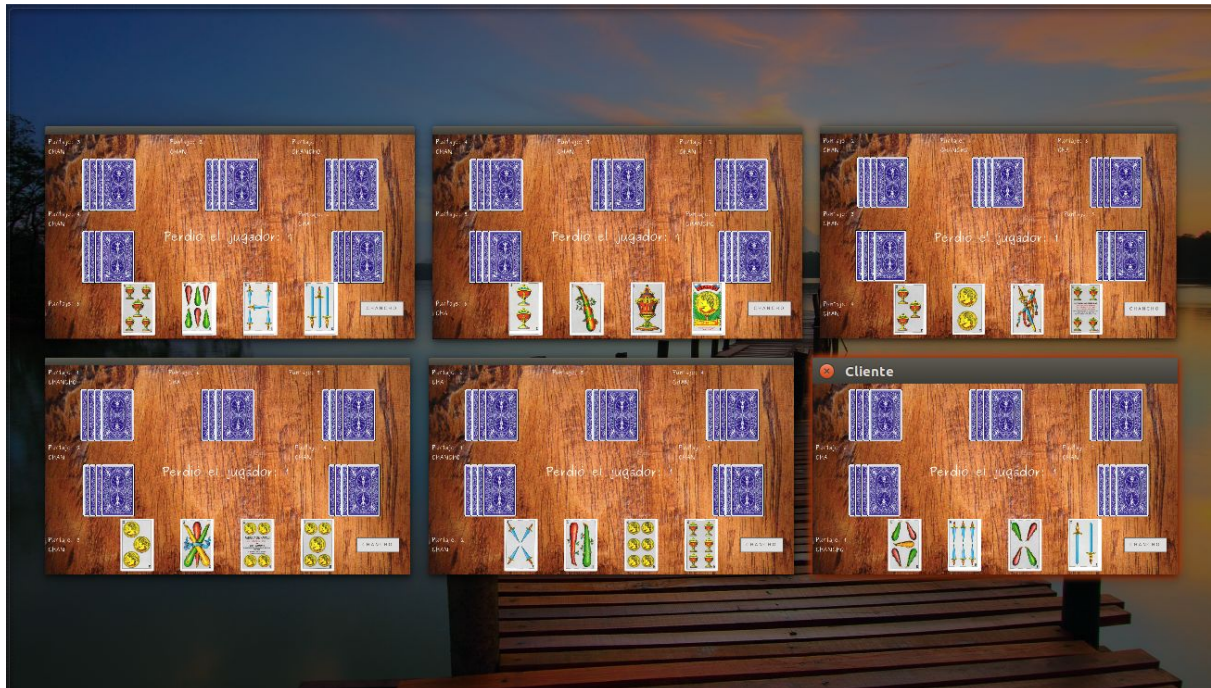
Tan pronto se conectan los seis jugadores, el servidor envía la señal para que se creen las interfaces de cada uno, de tal modo que cada jugador vea sus cuatro cartas y no la de los demás jugadores:



En la siguiente pantalla podemos ver como cuando un jugador presiona su botón CHANCHO, todos los otros jugadores pueden ver quién puso la mano, para así poder presionar el boton ellos también:



Además, cada jugador ve los puntajes que van sumando los otros jugadores y quién pierde al finalizar la partida:



Conclusión

Durante el desarrollo del trabajo integrador se aplicaron múltiples conocimientos adquiridos durante las clases de Programación Orientada a Objetos, tales como Clases, Objetos, Instanciación, Herencia, Agregación y Colecciones de Objetos, Polimorfismo, Streams de E/S, Constructores/Destructores, Sobrecarga de Métodos y Sobrecarga de Operadores.

Para el desarrollo de la interfaz de usuario se adoptó Qt por su licencia GNU, su amplia utilización y su completa documentación.

A partir del conocimiento obtenido durante la materia y el desarrollo del proyecto, se puede apreciar la importancia de las clases y objetos en el ámbito de la programación y la posibilidad de implementar de manera muy rápida el desarrollo de interfaces gráficas.