CardGame

By Victor Marition & Maxime Aubanel

I . Project Goal

Build a multiclients / server solution where the client represent a player and the server the game master. Then, implement a card game.

II . Our Project & its rule

We implement the **game battle**. Every client play a card each turn the client with the strongest card get 1 point. If there is a draw between 2 players or more, they get 1 point. Need more than 1 player to play.

III . Dependencies

We are using 2 external library : **NetworkCommsDotNet** (high level library to facilitate the using of socket and network) & **Newtonsoft.Json** (high level library which is very useful to serialize and deserialize object).

VI . Architecture

First of all, our project have been developed to make easy the implementation of another card game. The main function are generic and do not contain any object/data specific to the **game battle**.

a . Server

The server have 6 classes.

The server is split in two states. The lobby and the gameloop.

Users -> The User class contain data relatives to the user information.

Card -> The Card class contain data relatives to a card. Name, Value & Color.

Deck -> The deck class contain a list of Card and methods to work with it.

Server-> The Server class regroup all function relatives to the network.

Game -> The Game class regroup all function relatives to the GameLoop.

myNetwork_IP -> The myNetwork_IP class contain IP and Port of a User.

b . Client

The client have 3 classes.

Card -> The Card class contain data relatives to a card. Name, Value & Color.

Player -> The Player class contain a list of Card and methods for connection with server and GameLoop.

Program -> The Program class have all the callback's methods for smooth running of the game and main function

V . How To Play

- Launch the server.
- Launch severals client.
- Enter the IP of the server (the IP and Port are displayed on the server).
- Write “ready” or “notReady” (case sensitive) to update your status.
- When everyone in the lobby is ready, the game starts.
- Then, it’s pretty straightforward.

VI . Detailed class

a . Server class

The server class contain the function mandatory to the network mechanism. Thanks to the **NetworkCommsDotNet**, we are using its asynchrone function : AppendGlobalIncomingPacketHandler() to handle the different type of packet.

We have 6 types of packet :

Lobby	-> Used to update the client status.
Game	-> Used to get data from client ingame.
Message	-> Used to send message.
Ping	-> Ping.
Connection	-> Used to notify your first connection.
Start	-> Used when client is ready to play.

b . Game class

The game goes like this.

- Lobby();
- GameLoop();
    - DistribCard();
    - WaitUserReceiveCard();
    - Tick();
        - askUserToPlay();
        - WaitUserPlay(); // if this function return -2, the game is over
        - ProceedTurn();
        - RestartTurn();
    - SendUserWinner();
- ~Game();

The name of the function are explicit and describe exactly what they are doing.