

## System Programming (MEEC/MEAer)

### Project Assignment 2018/2019

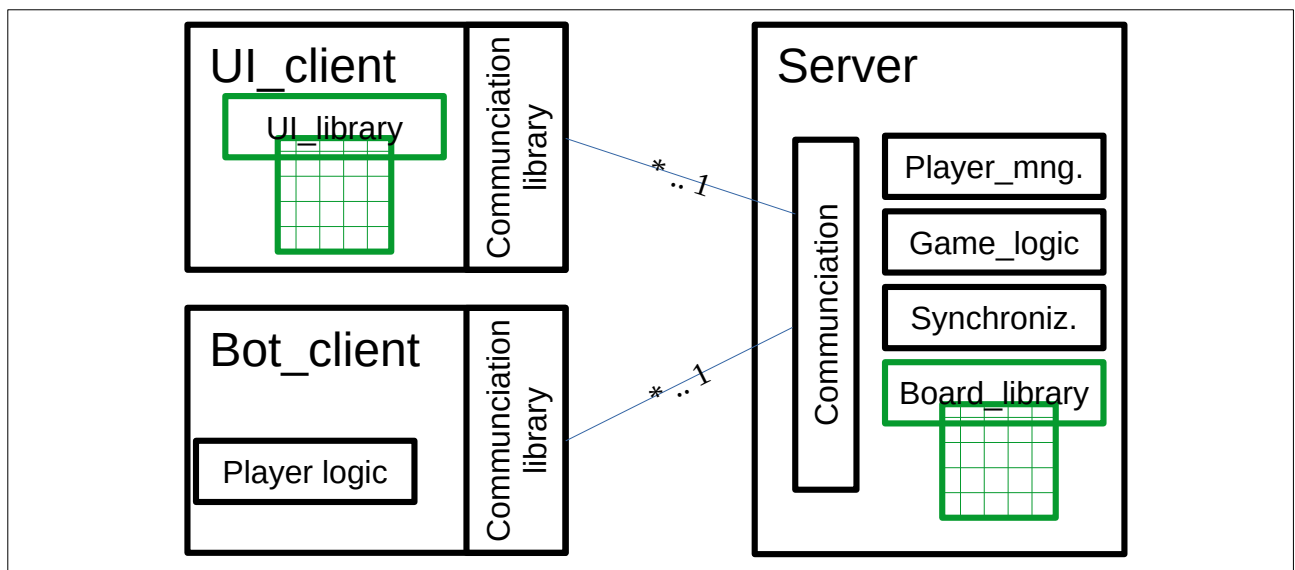
In this project the students will implement a simple distributed multiplayer concentration game ([https://en.wikipedia.org/wiki/Concentration\\_\(game\)\)](https://en.wikipedia.org/wiki/Concentration_(game))).

The game is composed of a board with a set of cards whose value are hidden from the players, for each card there is another one with the similar value. The main objective is to pick and pair cards with the same value.

When a player picks one card, it will be turned up and shown to every user, after a certain amount of time the card will be turned down. When a player select two cards with the same value, these cards will be kept shown until the end of the game. The game terminates when all cards are matched.

In order to allow multiple players student should implement a server that stores the board and receives the pick orders from the various users. The users need to execute a client, that will connect to the server and sends the pick order to the server. These clients will also show the state of the board (with shown and hidden cards). When a card is turned up or down all clients should be notified and the board representation should be updated.

Student should also implement a simple client (with no user interface) to automatically play.



## 1 Architecture

The main processes of the system are:

- Server
- UI clients
- Bot clients

In order to implement these main processes, students should use and develop the following modules:

- Board library
- UI library
- Communication library (for the clients)
- Player management code (on the server)
- Game logic (on the server)
- Synchronization (on the server and client)
- Player logic on the Bot

The Board library and UI library are provided by the teaching staff (shown in green on the next figure). These libraries contain the code to manage a board (initialization, check and pay validation) and present it in a graphical window. Students can change this code, but trying to maintain the same structure.

Multiple users or bots can be playing simultaneously against each other if connected to the same server.

## 2 Game rules



bc			ba
		baaa	
	aa		

The original game can use two decks of playing cards that are laid on a table facing down. In turns, each player chooses two cards and turns them up, if they are equal (same rank and color) then that player wins the pair and plays again.

The distributed version of the game will allow multiple user/players to play on a single board simultaneously. The players do not need to wait for their turn to pick a card on the board. All player will be able to pick cards simultaneously at any time.

Any change made and seen by one player (card turned up or down, or matched pair) will be forwarded to all other players.

The user should pick two cards in order to verify if they are equal. After picking two cards they will be compared. After picking two cards the player can continue playing.

When picking one card, it will be turned face up, and presented to all players. Simultaneous and concurrently other users can play by picking cards turned down.

## 2.1 Minimum number of players

The minimum number of player is 2. When the first player connects he will wait for another player to connect.

When there is only one connected player the game will freeze, until another player connects.

## 2.2 Player/user connection

When the first player connects to the server he will wait for another one.

When the second player connects, both players can start playing.

When a player connects during a game, he will:

- receive the state of the board (cards faced up)
- start receiving all updates (cards being turned dow or up)
- send picks to the server

Each player will be assigned a different color. This color is defined by the server.

## 2.3 Board

Each board is a square with DIMxDIM cards.

When the server starts, the DIM value is assigned from the command line (argv). All created boards will have this dimension.

Before a game is started the server initializes the board with the determined size and with random cards.

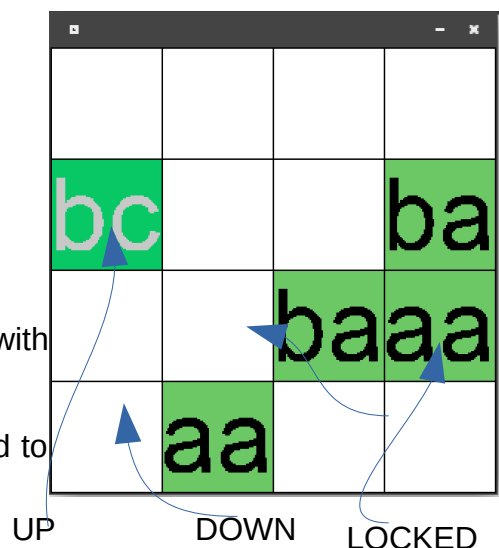
To simplify, each card contains a 2 characters string. Each string is assigned to a pair of cards.

## 2.4 Cards

Each card of the board can be in various states:

- DOWN – No user has picked it
- UP – one user picked it.
- LOCKED – Cards that were previously paired with other cards

When a user picks a card it is turned UP and assigned to that user. This card can not be picked while UP.



## 2.5 Card picks

The user can only pick DOWN cards.

The user is only allowed to pick two cards in sequence:

- The first pick card and the second pick card

### 2.5.1 First picks

The user can only perform the first pick when he does not have any UP card assigned to him.

After picking a card, the picked card it will be UP for 5 seconds.

During these 5 seconds the user can perform the second pick.

If the user does not perform the second pick these 5 second the card will be turned DOWN, and the user will be able to do a First pick.

### 2.5.2 Second picks

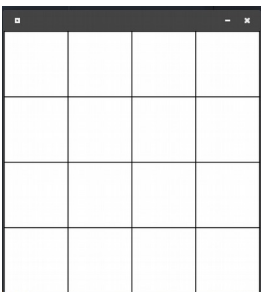
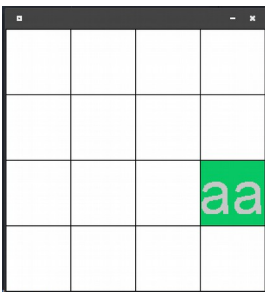
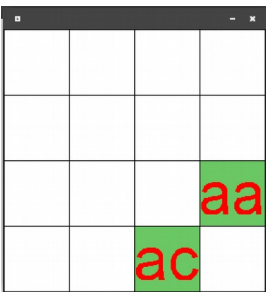
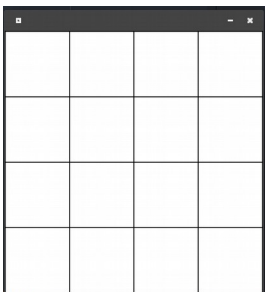
During the 5 seconds the first pick card is UP, the user can do the second pick.

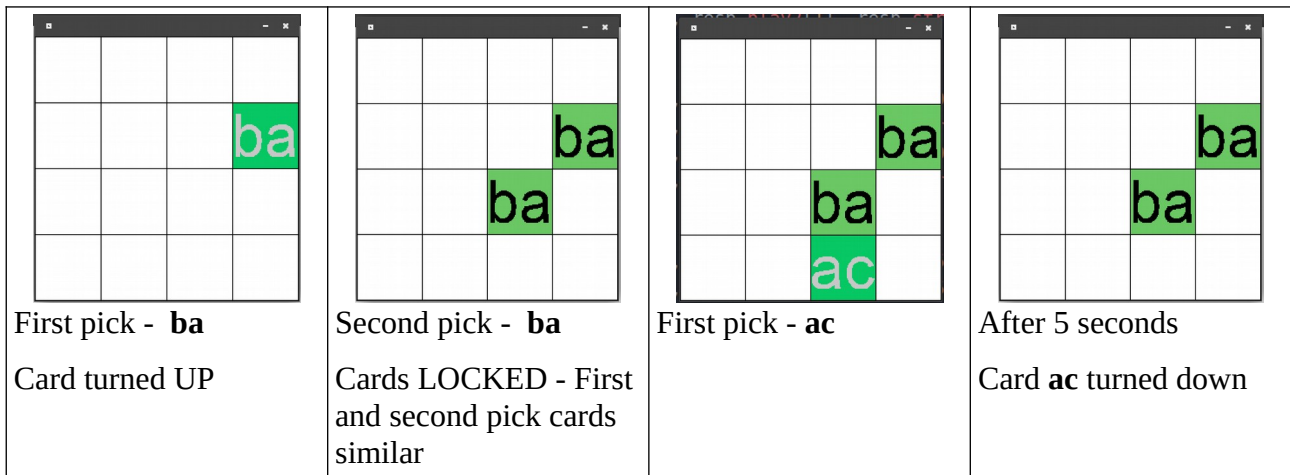
During this period, if the user picks an UP or LOCKED card, the first picked card will be turned DOWN. In this case the user will be able to immediately repeat a first pick.

When the user picks an DOWN card, this card will be turned UP.

If the two cards (first pick and second pick cards) are different, both will be kept UP for 2 seconds. After 2 seconds both cards will be turned DOWN.

If the two cards are equal, they will both be LOCKED. The user will be able to do a new first pick immediately.

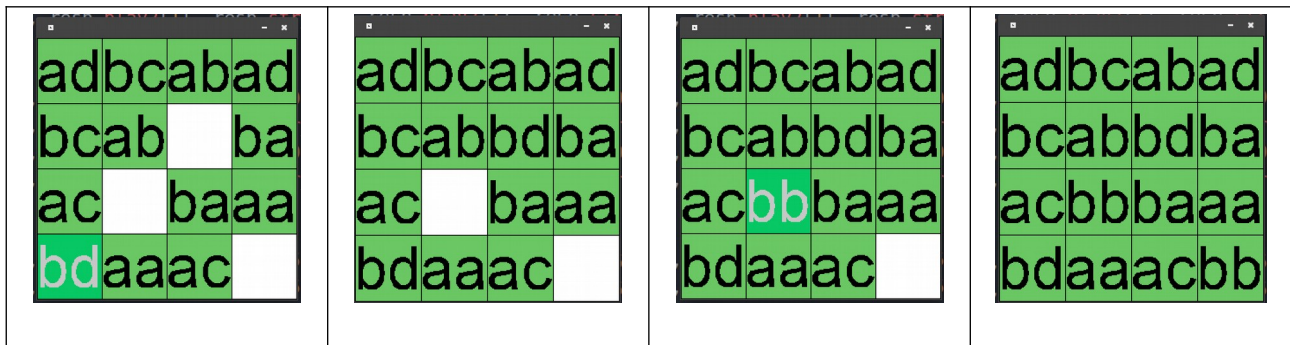
			
All cards DOWN	First pick - <b>aa</b> Card turned UP	Second pick – <b>ac</b> first and second pick cards different	After 2 seconds both card turn DOWN



## 2.6 End of Game

The game concludes when all cards have been paired and are in the LOCKED state.

The winner of the game is the player (or players) that paired more cards.



## 2.7 Multiple players

The system should also implement the following rules related to the possibility of multiple users playing on the same board.

### 2.7.1 Card picking

Each card can only be picked (UP) and paired (LOCKED) at a certain instance by one player. During the period of time that card can not be picked by any user.

If two users (each on his computer) pick the same card at the same time, the system has to guarantee that such card is only assigned to one of them.

### 2.7.2 Card colors

The background color depends on the user that last picked such card.

In the previous example there are 4 players picking cards:

- Blue Green yellow Pink

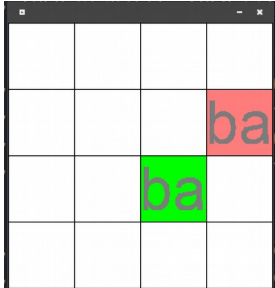
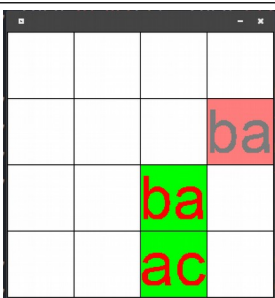
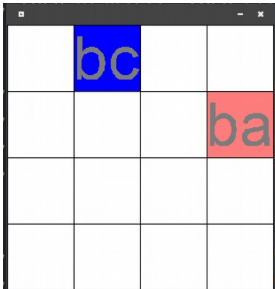
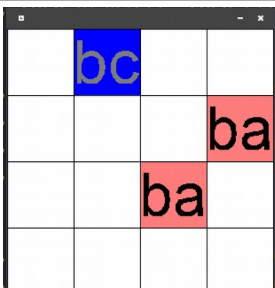
The system should assign a different color to each user.



All picks (card turned UP) or matches (cards LOCKED) done by a user should be represented by his color on all the players clients.

All cards turned DOWN are printed in white (with no text).

### 2.7.3 Text color

<b>First Pick</b>	
<p>The card corresponding to the first pick is represented with the text in gray.</p> <p>In this example players green and pink picked two cards</p>	
<b>Second Pick</b>	
<p>When the user picks a second card and both cards are different, the text color changes to red.</p> <p>In this example green player picked a different card from the first pick.</p>	
<p>After 2 seconds both cards picked by the green player are turned DOWN.</p> <p>Meanwhile the blue player picked a card (text in gray)</p>	
<p>The pink user waited for the green cards to be turned DOWN and selects the pair of its card.</p> <p>The second pick of pink user is similar to its first pick.</p> <p>These pink cards become LOCKED and the text is written in black.</p>	

### 3 Provided code

In order to implement this projects students should use a set o function to help the presentation of the boards and management of the board state.

These functions are provided in the board-library.c and UI\_library.c files. The necessary .h files are provided.

In order to compile the UI\_library, students should install the SDL2 and SDL2\_ttf Libraries

OpenSUSE	
SDL2	<a href="https://software.opensuse.org/package/libSDL2-2_0-0">https://software.opensuse.org/package/libSDL2-2_0-0</a>
SDL2_ttf	<a href="https://software.opensuse.org/package/libSDL2_ttf-2_0-0">https://software.opensuse.org/package/libSDL2_ttf-2_0-0</a>
Ubuntu	
SDL2	Package libsdl2-dev
SDL2_ttf	Package libsdl2-ttf-dev
MAC OS X / Windows (Development Libraries)	
SDL2	<a href="https://www.libsdl.org/download-2.0.php">https://www.libsdl.org/download-2.0.php</a>
SDL2_ttf	<a href="https://www.libsdl.org/projects/SDL_ttf/">https://www.libsdl.org/projects/SDL_ttf/</a>

The provided code uses the Arial Font. The True Type font file (arial.ttf) should be present in the same directory of the program. This file is is also supplied with the code.

**Before staring to develop the project students should guarantee that thay can compile the provided code example**

#### 3.1 Board library

The board\_library files allow the management of a single board game. This board is declared as a global variable in this file and is manipulated by the following functions. These functions represent a board (2 dimension matrix) as an array.

**void init\_board(int dim);**

The **init\_board** function creates a board with dim\*dim dimension and assigns each card a string. After the termination of this function the global variable corresponding to the board contains all the cards initialized.

**char \* get\_board\_place\_str(int i, int j);**

This function returns a pointer to the string of card referenced by the coordinates (i, j).

**play\_response board\_play (int x, int y);**

This function receives the coordinates of a card to be picked, and, depending of the card state (DOWN, UP, or LOCKED), updates it and returns suitable information.

## **3.2 UI library**

The functions available on the UI\_library.c allw the programmer to draw a board with a certain pixel dimension and number of cards. The window that will present the board and receive the clicks is declared as a global variable to this file, and all functions manipulate that variable.

**int create\_board\_window(int width, int height, int dim);**

this function creates a window with width\*height pixels and draw a board (with dim\*dim cards) with all cards turned DOWN.

**void close\_board\_windows();**

This function destroys the window that was used to present a board.

**void get\_board\_card(int mouse\_x, int mouse\_y, int \* board\_x, int \*board\_y);**

This function receive as arguments the mouse click coordinates (mouse\_x and int mouse\_y) and returns the corresponding board position (board\_x and board\_y).

The returned values can be used in the card drawing functions and in the board library.

**void write\_card(int board\_x, int board\_y, char \* text, int r, int g, int b);**

This function receives the card coordinates (board\_x and int board\_y) and print there the text. This function also receives as arguments the color of the TEXT (r, g, b arguments).

**void paint\_card(int board\_x, int board\_y , int r, int g, int b);**

This function receives the card coordinates (board\_x and int board\_y) and paints its background. This function also receives as arguments the color of the background to be painted (r, g, b arguments).

**void clear\_card(int board\_x, int board\_y);**

This function receives the card coordinates (board\_x and int board\_y) and clears it (paints it in white).



### 3.3 Example game

The provided application implements a single-user version of the project. The program:

- initializes the SDL2 libraries (SDL\_Init( SDL\_INIT\_VIDEO ) and TTF\_Init() )
- creates a window (create\_board\_window(300, 300, 4))
- initializes a board (init\_board(4))
- enters a loop for the reception of SDL events (quit, mouse down). When a user clicks with mouse the program:
  - converts mouse coordinates to board card location (get\_board\_card( ... )=
  - tries to pick a card (board\_play(board\_x, board\_y))
  - updates the board (switch (resp.code) { ... })

## 4 Server functionalities

The server allows clients to connect to play on a single board.

At a certain moment only one board is active and all the clients play on the same board.

The size of the board is defined at start by one of the program arguments (argv).

The server should create a socket and listen on all addresses on the port 3000.

The server should implement the game rules described previously.

The server should be multi-threaded.

When a game ends (all cards matched), all clients are notified about it. The winners receive information that they won.

After 10 second a new board is created and all the clients should create a new window.

From this moment on, the clients can start playing.

For debugging the server can show the evolution of the board in a graphical windows.

## 5 Clients

Students should implement two different clients that will interact with the server. Some of the code (communication layer) should be shared and reused between both applications.

### 5.1 UI Client

The UI Client is a program that connects to a server and allows a user to play against other users.

The UI Client receives as an argument the address of the server.

After receiving information about the size of the board, this client:

- creates on window
- receives the state of the board
- processes the SDL events
- receives the messages from the server

When a mouse click event is received from the SDL library this client should send a message to the server corresponding to a card pick.

When the client receives a message from the server, it must update the board window.

When a game ends, the client is notified, prints on the console the received information, and, after 10 seconds, creates a new board window to start the game.

## **5.2 Bot Client**

The Bot client is a program that connects to a server and plays automatically, without using any graphical element.

This program receives as an argument the address of the server, and starts sending picks to the server.

When the game ends it should print this information of the screen and start a new game.

The logic of this program should be programmed by the students and can range from simple random picks to a more complex heuristic with the storage of the board state.

This bot does not need to follow some of the rules imposed by the UI.