PROJECT GOMOKU REPORT

1. Project structure:

There are 7 header file:
- caro.h: declaration the main functions.

- graphic.h: declaration the colors and screens use in the program.

- data.h: declaration the save and load data function.

- GameState.h: declaration the game state of a gomoku game.

- machine.h: declaration the machine use in PVC mode.

- utils.h: declaration random, get current time and delay time function.

- ui.h: declaration user interface function.

And 8 source file:
- main.cpp: show how the program work;

- caro.cpp: definition the main functions.

- graphic.cpp: definition all functions in graphic.h

- data.cpp: definition all functions in data.h

- GameState.cpp: definition all function in GameState.h

- machine.cpp: definition all function in machine.h

- utils.h: definition all function in utils.h

- ui.cpp: definition all function in ui.h

In caro.h:
- load() : Initialize enviroment for ncurses and initialize variables.

- process(): run the program.

- close(): close all the program and free memory.

In graphic.h:

- In struct rectangle:
    + Function drawEdges() : draw edges around the rectangle.
    + Function drawTable(): draw table on the rectangel.

In namespace ObjectFall: animation of object fall from top to bottom

+ Struct Object: hold information of current position and icon of the objects.

+ Function update(): update position of current objects, erase objects run out of the field and add
new object.

+ Function erase(int x): erase object has index at x in object_set.

In namespace Graphic:

- Function init(): initialize variables and init namespace Screens, Color.

- In namespace Screens:

+ Function init(): initialize variables of screens.

+ Function load(): load data.

+ Function Clear(x, y, h, w): clear the rectangle ((x, y), (x + h – 1, y + w – 1)).

+ Function drawScreen(): draw current screen.

+ Function drawDemoStory(): draw demo story whille start the game.

+ Function drawMainScreen(): draw main screen in the game.

+ Function drawGameScreen(): draw play board, player information.

+ Function drawStatisticScreen(): draw statistic screen.

+ Function sketchStatisPVPScreen(): information of previous PVP matches.

+ Function sketchStatisPVCScreen(): information of previous PVC matches.

+ Function sketchOptionScreen(): draw option screen.

+ Function sketchSoundScreen(): change sound option.

+ Function sketchSizeScreen(): change size option.

+ Function sketchIconScreen(): change player icon option.

+ Function updatePtr(id, x): update the cursor by subtracting x or adding x of screen ID.

+ Function getPtr(id): return the position of cursor of screen ID.

+ Function updateCurrentScreen(x): change the current screen to screen x.

+ Function getCurrentScreen(): return the id of current screen.

- In namespace Color:
    + Function init(): initialize variables and color.

    + Function setBackgroundColor(x): change background color to x.

    + Function colorOn(x): turn on the color x for all characters after calling this function.

    + Function colorOff(x): turn off the color x for all characters aftet calling this function.

    + Function reverseColorOn(x): turn on the color x and reverse background color for all character
    after calling this function.

    + Function reverseColorOff(x): turn off the color x and reverse background color for all character after calling this function.

    + Function reverseOn(): turn on reverse background color of all character after calling this function.

    + Function reverseOff(): turn off reverse background color of all character after calling this function.

- In data.h:
Namespace Data:
    + Namespace Save:
        * Function savedGame(flag): save data of current match to file.

        * Function canLoadGame(flag): check if we can load old game from file.

        * Function loadGame(flag): load old match from file.

    + Namespace Statis:

        * Function saveGame(flag): save win match to statistic.

        * Function loadStatis(flag): load data from statistic file.

        * Function getStatisSize(): get the size of the saved game from statistic file.

        * getStatisName(i): return the date time of match at index I.

        * to_int(s): convert string to integer.

- In GameState.h:
    + Namespace playerState: information of the player.

    * init(x, y, h, w, cur, _color, _chess, _name): (x, y, w, h): the position of rectangle where print information of player, color of player chess is _color, character of player chess is _chess, name of player is _name.

    * setColor(x): change player chess color to x.

    * setName(_name): change player name to _name.

    * setChess(c): change player chess character to c.

    * getIcon(): get player chess character.

    * getName(): get player name.

    * doMove(x, y): player take turn at ceil (x, y).

    * next(): player finish his turn and change to next player.

    * isWinner(): change variables when player wins a match.

    * IsLoser(): chang variables when player loses a match.

    * isDraw(): chang variables when player get draw in a match.

    * printProfile(): print player profile to screen.

    * resetProfile(): reset player profile.

- setup(m, n): initialize variables and set the play board to m x n.

- setTypeGame(x): get the type of match to x(0: PVP match, 1: PVC match).

- setStateRow(x): change the height of board to x.

- setStateCol(x): change the width of board to x.

- setStateAt(x, y, v): change the value of ceil at (x, y) to v.

- setBoardSize(x, y): change the board size to x * y.

- getTypeGame(): return the type of match.

- getStateRow(): return the height of board.

- getStateCol(): return the width of board.

- getStateAt(x, y): return value at (x, y).

- getBoardHeight(): return the rectangle height.

- getBoardWidth(): return the rectangle width.

- getPtrOx(): return the vertical position of current cursor.

- getPtrOy(): return the horizontal position of current cursor.

- print(): print the play board and player information.

- reset(flag): reset the variables.

- doMove(): current player take turn.

- machine(std::pair <int, int>): if there is a PVC match the machine will take turn.

- nextTurn(): change turn to next player.

- haveWinner(): check if the match have winner or not.

- canMove(): check if player can take turn or not.

- updateData(): when the match end, update the player data.

- backToMainScreen(): when user want to back to main screen, run this function.

- undoProcess(): undo.

In machine.h:

- In namespace Machine:
    + check(x, y, _icon): calculate the cost if player _icon take turn at (x, y).

    + getMove(): return the position where the machine will choose.

    + doMove(): machine take turn.

In utils.h:

- In namespace Utils::Random:

    + seed(): call this function befrore call get(l, r).

    + get(l, r): return random value between (l, r).

In ui.h:

- In namespace Input: Contain function to check when user input and character.
    + void read();

    + bool isArrowKey();

    + bool isKeyLeft();

+ bool isKeyRight();

+ bool isKeyUp();

+ bool isKeyDown();

+ bool isEnterKey();

+ bool isEscKey();

+ bool is_Q_Key();

+ bool is_R_Key();

+ bool is_S_Key();

+ bool is_Z_Key();

+ bool is_U_Key();

+ bool is_G_Key();

+ bool is_C_Key();

+ bool is_N_Key();

+ bool is_B_Key();

+ bool isDigit();

+ bool isAlpha();

+ int getInput();

- In namespace Controller: Contain the function process the input.

+ void process();

+ void arrowKeyProcess();

+ void keyUpProcess();

+ void keyDownProcess();

+ void keyLeftProcess();

+ void keyRightProcess();

+ void enterKeyProcess();

+ void backSpaceKeyProcess();

+ void PvPStatisControl();

+ void PvCStatisControl();

+ void undoProcess();

+ void soundControl();

+ void sizeControl();

+ void iconControl();

+ int makeSound();

+ void setSound(int x);

2. Data structure:

- Use some data structure in STL library: map, vector, pair <int, int>.
- Machine in PVC mode work with Greedy algorithm: calculate the cost of the ceil(x, y) and get the ceil has maximum cost.

3. Remarks:

- In save game: file must be save in this format:

```
/** saved*.game format
 * flag (0 : do not exist; 1 : exist)
 * type (0 : PVP; 1 : PVC)
 * m n  (m : number of rows; n : number of cols)
 * M lines: each line n numbers. Represent the gameboard of the match.
 *      -> 1 : first player;
 *      -> 2 : second player/machine;
 *      -> 0 : no one do in this cell.
 * */
```

- In statistic: statistic file must be save in this format:

```
/** *.statis format
 * Consist of number of lines.
 * Each line has: yyyy.mm.dd hh:mm:ss m n state
 *
 * Note:
 *  + yyyy.mm.dd : year.month.day of the play;.
 *  + hh:mm:ss time when end the game.
 *  + m, n: length of lines and cols of the state.
 *  + state: game board of the match
 * */
```

- Algorithm use for machine in PVC mode:

    + For each ceil (x, y), calculate the cost if machine take turn on this ceil and player take turn on this ceil.

+ If the cost is really huge so that there are to much case machine can’t handle, ignore this ceil. Otherwise, push this ceil in a list let call it A.

+ Sort list A descending order of cost and take the first ceil.

- We can use this algorithm along with Minimax or Monte Carlo Tree Search to improve the result.