Report of Caro's Project

ALL ABOUT PROJECT STRUCTURE AND DATA STUCTURE

GROUP 7 – 18 CTT3 | CS 161 | November 11, 2018

No.	Student ID	Full Name	Percentage of Contribution
1	18125088	NGUYEN LE THANH KHIET	15%
2	18125091	NGUYEN MINH KHUE	25%
3	18125092	NGUYEN TRAN MINH KHUE	30%
4	18125129	NGUYEN TRUNG HAU	30%

I. ALL TASKS HAD DONE:

1. Splash Screen & About Us
2. PvP
3. PvC
4. Save/Load Game
5. Option

All tasks had done:

1.	Discuss with the group	(2 days)
2.	Read idea of other member's code	(3 days)
3.	Research to add file header, file cpp	(1 day)
4.	Research function	(2 days)
5.	Research string and array	(2 days)
6.	Research structures	(2 days)
7.	Research draw console board	(1 day)
8.	Research control by target buttons	(2 days)
9.	Research change color word	(1 hour)
10.	Research read external file	(2 days)
11.	Research AI (PvC)	(2 days)
12.	Idea for about screen	(1 day)
13.	Code about screen	(2 hours)
14.	Idea for Menu screen	(1 day)
15.	Code Menu screen	(2 hours)
16.	Ideal for PlayPvC mode	(5 days)
17.	Feedback with the group	(2 hours)
18.	Do report	(4 hours)

All tasks had done:

1.	Read part of work of other member ‘code.	(3 days)
2.	Research to add file header , file cpp.	(2 hours)
3.	Read ideal of the LoadPvP, SaveGamePvP.	(1 day)
4.	Run the program of PvP.	(1 hour)
5.	Find the direction of icons X,O.	(1 day)
6.	Find the direction for LoadPvC.	(2 days)
7.	Edit the code of LoadPvC.	(2 days)
8.	Edit the code of SaveGamePvC .	(2 days)
9.	Code new program for LoadPvC.	(2 hours)
10.	Code new program for SaveGamePvC .	(2 hours)
11.	Correct some mistakes for running all.	(2 days)
12.	Running for all program .	(1 hour)
13.	Check all program.	(1 hour)
14.	Research structure.	(1 day)
15.	Do report.	(2 days)

All tasks had done:

1.	Research about a caro game	(3 hours)
2.	Research about functions	(1 hour)
3.	Research how to move the cursor (GoToXY)	(1 hour)
4.	Research how to move with arrows	(2 hours)
5.	Research about how to draw frames	(a day)
6.	Search ASCII code for draw frame	(30 mins)
7.	Research how to change text color	(1 hour)
8.	Search how to exit	(30 mins)
9.	Search how to delay	(1 hour)
10.	Give tasks to other members	(1 day)
11.	Edit SplashScreen	(1/2 day)
12.	Search how to turn on/off the cursor	(30 mins)
13.	Edit AboutUs	(1/2 day)
14.	Research about dividing files	(1 hour)
15.	Research about how to draw a caro board	(a day)
16.	Research about 2-dimension array	(3 hours)
17.	Research about Win – Draw Rule	(1/2 week)
18.	Thinking how to return to menu	(1/2 day)
19.	Code PvP	(3 days)
20.	Debug, edit PvC	(a day)
21.	Think how to resize a gameboard	(1/2 day)
22.	Research about save a game	(a day)
23.	Thinking of what to save	(a day)
24.	Code Save File PvP	(a day)
25.	Code Load Game PvP	(1/2 day)
26.	Research how to load different types of save files	(a day)
27.	Edit Load Game	(a day)
28.	Explain to members about idea of option	(3 days)
29.	Summarize all reports	(2 days)

All tasks had done:

1. ideal of other member's code (2 days)
2. Research to add file header, file cpp (1 day)
3. Research function GoToXY(x,y); (1 day)
4. Find icons for X, O (1 hour)
5. Research draw console board (1 day)
6. Research control by target buttons (2 days)
7. Research change color word (1 hour)
8. Research read external file (2 days)
9. Ideal for change icon (1 day)
10. Code change icon (1 day)
11. Ideal for change size game board (1 day)
12. Code change size game board (1 day)
13. Ideal for change game rule (1 day)
14. Code change game rule (1 day)
15. Ideal for LoadwinPvP (2 days)
16. Code loadwinPvP (2 days)
17. Ideal for loadwinPvC (2 days)
18. Code loadwinPvC (2 days)
19. Research structure (1 day)
20. Do report (3 hours)

II. PROJECT STRUCTURE:

1. FILE SOURCE AND HEADER:

HEADER FILE	SOURCE FILE
• Console.h: contains declaration library and prototype of functions relating to adjusting the console	• Console.cpp: contains functions relating to adjusting the console
• Model.h: contains declaration library and prototype of functions relating to rules/tools/bases supporting the game/check win	• Model.cpp: contains functions relating to rules/tools/bases supporting the game/check win
• Controller.h: contains all the library for project	
• View.h: contains declaration library and prototype of functions relating to show splash sceen/menu/introduction	• View.cpp: contains functions relating to show splash screen/menu/introduction
• Draw.h: contains declaration library and prototype of functions relating to draw frame/choose buttons	• Draw.cpp: contains functions relating to draw frame/choose buttons
• GameController.h: contains declaration library and prototype of functions relating to playing/loading games	• GameController.cpp: contains functions relating to controlling playing/loading games
• PlayMode.h: contains declaration library and prototype of functions relating to play PvP/PvC	• PlayMode.cpp: contains functions relating to play PvP/PvC
• OptionController.h: contains declaration library and prototype of functions relating to Options	• OptionController.cpp: contains functions relating to Options
• Point.h: contains structure supporting position controlling	• main.cpp: play game

2. MEANING OF FUNCTIONS:

❖ Console:

- `void GoToXY(int x, int y):` Move the cursor to coordinate(x,y).

  Input: 2 integers x (for horizontal axis) and y (for vertical axis)

  Output: none

- `void ShowConsoleCursor(bool show):` Turn on/off the console cursor

  Input: `true/false`

  Output: none

- `void TextColor(int colorCode):` Change the color of the words.

  Input: integer performs the color:

Brown	6
Light Cyan	11
Light Red	12
Yellow	14
White	15

  Output: none

❖ Draw:

- `void DrawChooseMapMenu():` Draw the Choose Map Menu (3x3, 5x5, 10x10)

  Input: none

  Output: none

- `void DrawChooseMode():` Draw the Mode Map

  Input: none

  Output: none

- `void DrawChooseModeLoadGame`(): Draw the Load Game Map

  Input: none

  Output: none

- `void TargetButton(int whichButton, bool on): Move to the` target button

  Input: an integer performs target button (0/1/2/3/4), true/false

  Output: none

- `void DrawFrame(Point positionToDraw, int length, int height, char text[]):` Draw the Frame

  Input: a structure Point performs position (x, y) , integer performs the length of Frame, integer performs height of frame, and string text performs the text in frame

  Output: none

- `void RowVertical(int size):` Print vertical line each row

  Input: integer performs size of game board (3/5/10)

  Output: none

- `void RowConnerandEdge(int size, int left, int middle, int right):` Print horizontal line each row

  Input: integer performs size of game board (3/5/10), integer performs the number of ascii code for lines (218/192/195), 194/193/197, 191/217/180

  Output: none

- `void PrintRow(int size, int num, int count):` Control printing lines of map

  Input: 3/5/10, number of rows ([0 , size]), number of rows ([0 , size])

  Output: none

- `void PrintMap(int size):` Print map and fill all the blank with " "
  Input: 3/5/10
  Output: none

- `void DrawMap(int size, char map[15][15]):` Draw the Map

  Input: integer performs size of game board (3/5/10), map

  Output: none

❖ Model.

- `int GetChooseButton(int numberOfButtons):` Receiving the keyboard hit to calculate which button is hit and draw the frame of the one being hit

  Input: numbers of buttons.

  Output: number of button being hit

- `bool CheckWin(char map[15][15], int size, Point position):` Check win the game without special rule

  Input: map, 3/5/10, position (x,y)

  Output: true/false

- `bool CheckDraw(char map[15][15], int size):` Check draw the game

  Input: map, 3/5/10

  Output: true/false

- `bool CheckWinOption(char map[15][15], int size, Point position):` Check win the game with special rule

  Input: map, 3/5/10, position (x,y)

  Output: true/false

- `void ReadListSaveFiles1(char listFiles[50][50], int & countFiles):` Count files, find files with .s in the end

  Input: listSaveFiles, integer performs saved game countFiles

  `Output: none`

- `void ReadListSaveFiles2(char listFiles[50][50], int & countFiles):` Count files, find files with .m in the end

  `Input:` listSaveFiles, integer performs saved game countFiles

  `Output:` none

- `void ReadListWinFiles(char listFiles[50][50], int & countFiles):` Count files, find files with .w in the end

  `Input:` listSaveFiles, integer performs saved game countFiles

  `Output`: none

- `void ReadListWinFiles2(char listFiles[50][50], int & countFiles):` Count files, find files with .d in the end

  `Input`: listSaveFiles, integer performs saved game countFiles

  `Output:` none

❖ Gamecontroller

- `int ChooseFirstPlayer(): Random which player will go first`
  Input: none
  Output: 1/2

- `void NewGame(int& Xicon, int& Oicon, int& win, int& sizeboard, int& SttPP, int& gamewinPP, int& gamedrawPP, int& SttPC, int& gamewinPC, int& gamedrawPC):` Play a new game.
  Input: Integers:

Xicon (default = 88) Oicon (default =79)	Xicon, Oicon: (88 / 79 / 145 / 153 / 157 / 158 / 224) the numbers of characters in ASCII.
Win (default = 1)	Win: integer(1 or 0) performs rule game.
Sizeboard (default =3)	Sizeboard: (3or 5 or10) Performs size of board.
SttPP (default =0)	SttPP: counts game that is played in PvP mode.
gamewinPP (default = 0)	gamewinPP: counts game that player1 win in PvP mode.
gamedrawPP (default = 0)	gamedrawPP: counts game that draw in PvP mode.
SttPC (default =0)	SttPC: counts game thay is played in PvC mode.
gamewinPC (default =0)	gamewinPC: counts game that player win.
gamedrawPC (default =0)	gamedrawPC: counts game that draw in PvC mode.

  Output: none

- `void ChooseModeLoadGame(int& Xicon, int& Oicon, int& win, int& SttPP, int& gamewinPP, int& gamedrawPP, int& SttPC, int& gamewinPC, int& gamedrawPC):` Choose mode to load game
  Input: Integers :

Xicon (default = 88)	Xicon, Oicon: (88 / 79 / 145 / 153 / 157 / 158 / 224) the numbers of characters in ASCII.

Oicon (default =79) Win (default = 1) Sizeboard (default =3) SttPP (default =0) gamewinPP (default = 0) gamedrawPP (default = 0) SttPC (default =0) gamewinPC (default =0) gamedrawPC (default =0)	Win: integer(1 or 0) performs rule game. Sizeboard: (3or 5 or10) Performs size of board. SttPP: counts game that is played in PvP mode. gamewinPP: counts game that player1 win in PvP mode. gamedrawPP: counts game that draw in PvP mode. SttPC: counts game thay is played in PvC mode. gamewinPC: counts game that player win. gamedrawPC: counts game that draw in PvC mode.

Output: none

- `void LoadGamePvP(int& Xicon, int& Oicon, int& win, int& SttPP, int& gamewinPP, int& gamedrawPP)`: Load the GamePvP

  Input: Integers

Xicon (default = 88) Oicon (default =79) Win (default = 1) Sizeboard (default =3) SttPP (default =0) gamewinPP (default = 0) gamedrawPP (default = 0) SttPC (default =0) gamewinPC (default =0) gamedrawPC (default =0)	Xicon, Oicon: (88 / 79 / 145 / 153 / 157 / 158 / 224) the numbers of characters in ASCII. Win: integer(1 or 0) performs rule game. Sizeboard: (3or 5 or10) Performs size of board. SttPP: counts game that is played in PvP mode. gamewinPP: counts game that player1 win in PvP mode. gamedrawPP: counts game that draw in PvP mode. SttPC: counts game thay is played in PvC mode. gamewinPC: counts game that player win. gamedrawPC: counts game that draw in PvC mode.

Output: none

- `void LoadGamePvC(int& Xicon, int& Oicon, int& win, int&SttPc, int& gamewinPC, int& gamedrawPC): Load the PvC's Game`

Input: Integers

Xicon (default = 88) Oicon (default =79) Win (default = 1) Sizeboard (default =3) SttPP (default =0) gamewinPP (default = 0) gamedrawPP (default = 0) SttPC (default =0) gamewinPC (default =0) gamedrawPC (default =0)	Xicon, Oicon: (88 / 79 / 145 / 153 / 157 / 158 / 224) the numbers of characters in ASCII. Win: integer(1 or 0) performs rule game. Sizeboard: (3or 5 or10) Performs size of board. SttPP: counts game that is played in PvP mode. gamewinPP: counts game that player1 win in PvP mode. gamedrawPP: counts game that draw in PvP mode. SttPC: counts game thay is played in PvC mode. gamewinPC: counts game that player win. gamedrawPC: counts game that draw in PvC mode.

Output: none

❖ PlayMode:

- `int PvP(char map[15][15], int first, int size, int Xicon, int Oicon, int win, int& SttPP, int& gamewinPP, int& gamedrawPP): Run PvP game`

Input: map, 1/2 , 3/5/10, 88/145/153/157/158/224 , 79/145/153/157/158/224, 0/1, numbers of played games, numbers of win games, numbers of draw games
Output: 0/1/2

- `int PvC(char map[15][15], int first, int size, int Xicon, int Oicon, int win, int& SttPC, int& gamewinPC, int& gamedrawPC): Run the PvC game`
- *Input:* map, 1/2 , 3/5/10, 88/145/153/157/158/224 , 79/145/153/157/158/224, 0/1, numbers of played games, numbers of win games, numbers of draw games
  Output: 0/1/2

❖ View

- `void ShowSplashScreen(): show the Splashscreen.`

  Input: none

  Output: none

- `void ShowStartMenu(): show Start Menu.`

  Input: none

  Output: none

- `void AboutUs(): Show the introduction`

  Input: none

  Output: none

❖ Option

- `int option():` Ask Player for a choice.

Input: none

Output: An integer that performs a choice.

- `void ControllOption(int& Xicon, int& Oicon, int& win, int& sizeboard, int& SttPP, int& gamewinPP, int& gamedrawPP, int& SttPC, int& gamewinPC, int& gamedrawPC)`: Control choices in option.

Input: Integers :

Xicon (default = 88) Oicon (default =79)	Xicon, Oicon: (88 / 79 / 145 / 153 / 157 / 158 / 224) the numbers of characters in ASCII.
Win (default = 1)	Win: integer(1 or 0) performs rule game.
Sizeboard (default =3)	Sizeboard: (3or 5 or10) Performs size of board.
SttPP (default =0)	SttPP: counts game that is played in PvP mode.
gamewinPP (default = 0)	gamewinPP: counts game that player1 win in PvP mode.
gamedrawPP (default = 0)	gamedrawPP: counts game that draw in PvP mode.
SttPC (default =0)	SttPC: counts game thay is played in PvC mode.
gamewinPC (default =0)	gamewinPC: counts game that player win.
gamedrawPC (default =0)	gamedrawPC: counts game that draw in PvC mode.

Output: none

- `void ChangeAllIcon(int& Xicon, int& Oicon, int& win)`: Change Icon X and O.

Input : Integers:

Xicon (default = 88): 88/145/153/157/158/224

Oicon (default =79): 79/145/153/157/158/224

Win (0/1): Rule of check win.

Output: none

- `void ChangeSIZEgameboard(int& sizeboard):` Change size of game board.

  Input: 3/5/10
  Output: none

- `void ChangeRule(int& win):` To change rule for check win.

  Input : integer win (default =1) for check win.
  Output: none

- `void LoadWinPvP():` To load game which is saved in PvP mode.

  Input : none
  Output: none

- `void LoadWinPvC():` To load game which is saved in PvC mode.

  Input : none
  Output: none

- `void Statistic(int SttPP, int gamewinPP, int gamedrawPP, int SttPC, int gamewinPC, int gamedrawPC):` Save history of game.

  Input:
  SttPP (default =0),
  gamewinPP (default = 0),
  gamedrawPP (default = 0),
  SttPC (default =0),
  gamewinPC (default =0),
  gamedrawPC (default =0).
  Output: none

III. DATA STRUCTURE:

- Example of a saved game:

3 2

X N N

O X N

N O X

3: size of current gameboard

2: turn of the current player

X, O: checked cells

N: blank cells

- Description of how to save a game: We save the size of the current gameboard, then save " ", then save the turn of the current player, finally the 2-dimension array, if the cell is blanked, save it as "N"