* **Introduction**

The purpose of writing this program is to make a chess battle platform with both graphic version and text version which can accommodate both human and computer players. Computer player has four different levels. Human players can choose different computer levels to change the difficulty of the game. Besides, we also provide players with a setup mode outside the game, in which players can build a board of their own and use it to solve some chess endgame.

* **Overview**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Breakdown of the Project** | **Estimated Completion Dates** | **Actual Completion Date** | **Group Members**  **Respond** | |
| Drawing the first vision of UML | 22nd March | 22nd March | Zhijie Yu (z224yu) |  |
| Polishing the first vision of plan of attack | 24th March | 24th March | Zhijie Yu (z224yu) | Yue Yu (y329yu) |
| Six kinds of pieces and Subject Class | 26th March | 28th March |  | Yue Yu (y329yu) |
| Observer Pattern and Textdisplay Class | 26th March | 29th March | Zhijie Yu (z224yu) |  |
| Grid Class | 27th March | 31st March | Zhijie Yu (z224yu) | Yue Yu (y329yu) |
| Main function and Human class to play the game | 27th March | 3rd April | Zhijie Yu (z224yu) |  |
| Computer Level 1 and Level 2 Class | 29th March | 2nd April | Zhijie Yu (z224yu) |  |
| GraphicsDisplay Class | 29th March | 4th April | Zhijie Yu (z224yu) |  |
| Computer Class Level 3 | 30th March | 3rd April |  | YuYu ( y329yu) |
| Computer Level 4 Class | 31th March | Not finished | Zhijie Yu (z224yu) | Yue Yu (y329yu) |
| Update final UML | 1st April | 3rd April | Zhijie Yu (z224yu) |  |
| Final design document | 1st April | 4th April |  | Yue Yu (y329yu) |
| Extra features | 3rd April | 4th April | Zhijie Yu (z224yu) | Yue Yu (y329yu) |

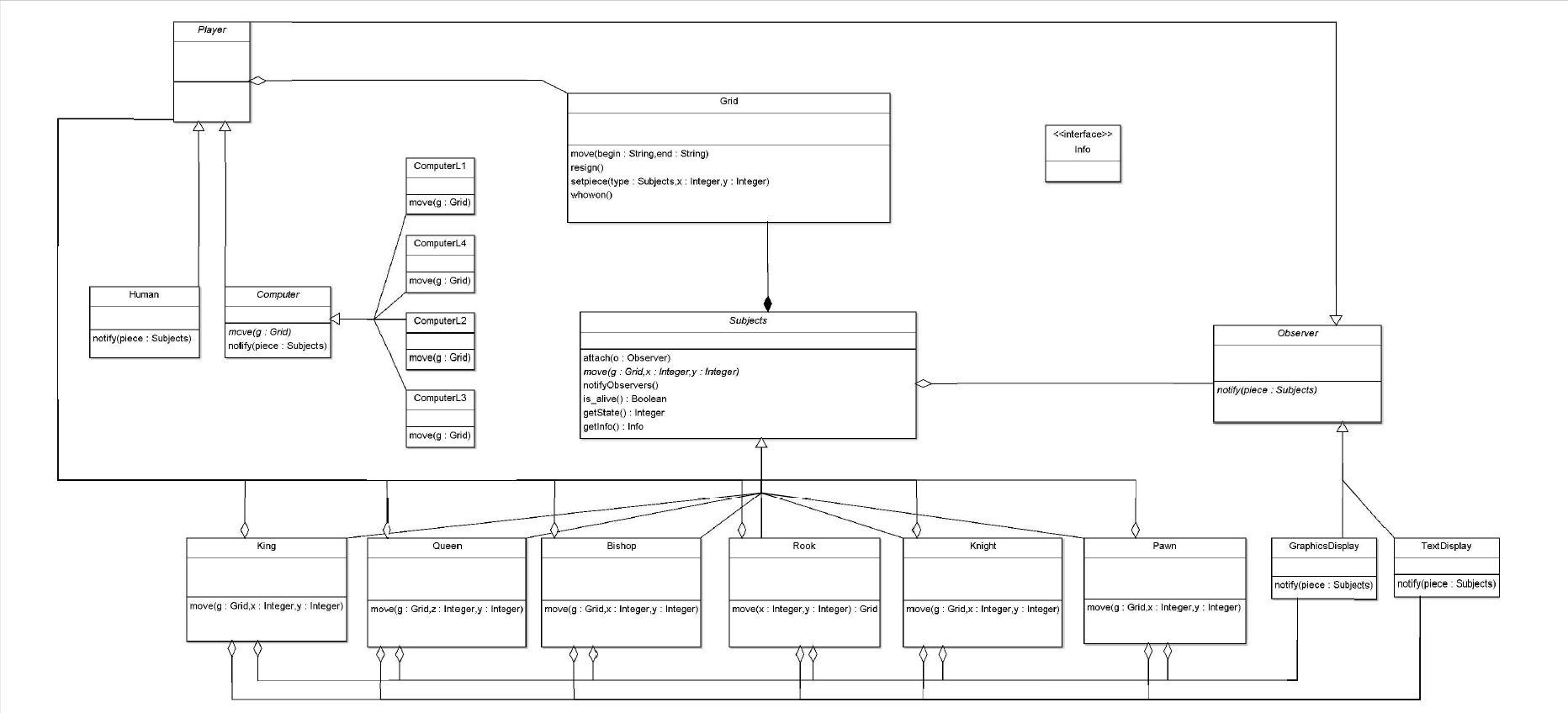
Since it is a chess game, we need class Piece, class Slot and class Grid. In the Piece, there are six kinds of subclasses which are King, Queen, Bishop, Rook, Knight and Pawn. In every piece, we have the corresponding coordinate (int x and int y), the ownership and a method move to check whether the piece can move to a specific position. Particularly, we have is\_checking method in King to check whether the king is being checked. Then, we wrap every Piece in a corresponding Slot which has a pointer to a Piece. In a Slot, we can put (replace), get or put\_back a Piece into a Slot. Next, in Grid, we have a vector of vector of Slot from (0, 0) to (7, 7) and also vector of class Player. We can move Piece, remove Piece, undo moves, get the information of a specific Piece, print a text grid, and know whether either player is being checked in a Grid. Besides, we use class Human to control game between human and use computer1, computer2, computer3 to control game with or within different levels computer. In main function,

In the Grid c, we have vector of vector of class Slot from (0, 0) to (7, 7). Every Slot is a wrapper of a pointer to a Pieces or nullptr. In a Slot, we can

We use a class to record the previous move. When the current move is a Pawn; and the Pawn is capturing on the forward diagonals, we check whether the previous move of opponent is moving a Pawn from start-line two squares. If so, then compare the start-point and the end-point of my Pawn with the end-point of previous move of the opponent’s move. If it corresponding to the rule, the opponent’s Pawn can be captured.

In computer[3], we put pieces with different ownership into different vectors. Then, we calculate every slot is under which player’s attack, e.g. if two player1’s pieces can move to a specific slot in one move, and 3 player2’s pieces can move to the same slot in one move, this slot is under player2’s control. We put pieces under different player’s control into different vector of position. When computer make decision of its move, it prefers to move its pieces from the position under others control to its own control. Then it can avoiding capture.

In computer[4], we sort the order of the vector of position under both player’s control in computer[3], by comparing the weight of the piece on the position. The weight of Queen is 9, Rook is 5, Knight is 3, Bishop is 3, and Pawn is 1. The computer prefers to move pieces with higher weight to avoiding capture. Besides, when the computer do not need to avoiding capture, it should move to some position with high significance. The position which is much closer to the center is much more significant, because on this position, the piece is much flexible.

* **Update UML**

**Answers** **to Questions**

* Answer to question 1:

First we will download different popular standard openings and rewrite each step into a vector of command *move* e.g. *vector <e2, e4> first\_move*. Then store each standard opening in a vector of vector of *move* e.g.

*vector <vector <move, move>>* *Italian\_Game;*

*Italian\_Game[0] = <e2, e4>;*

*Italian\_Game[1] = <e7, e5>;*

*Italian\_Game[2] = <g1, f3>;*

*Italian\_Game[3] = <b8, c6>;*

*Italian\_Game[4] = <f1, c4>;*

*……*

We also use a class to store all standard openings we have e.g.

*class st\_opening{*

*vector <vector <move, move>>* *Italian\_Game;*

*vector <vector <move, move>> French\_Defense*

*vector <vector <move, move>> Foue\_Knights\_Game;*

*……*

*};*

Next, we provide a feature called standard\_opening which can control whether a standard opening will be used or not and which standard opening will be used by computer. This feature can be used only when computer vs. computer and human vs. computer because it will affect the moves of computer. When player chooses one of the standard openings during human vs. computer, the computer will give priority to the corresponding movement, but not always follow the rules. We will include another algorithm to determine the moves of computer. When players’ move corresponds to the standard opening, computer will follow the rule. However, when player break the standard opening or the standard opening has finished, computer will try to find the best move according to current situation. When it is not being captured or checked and cannot capture opponent, it will still follow the standard opening. Otherwise, it will break the standard opening and choose the best move. As long as it breaks the standard opening, it will not follow it again. During computer vs. computer, both computers will follow the chose standard opening until it finishes. When the control feature is not be used, computer will check automatically whether its opponent is using a standard opening or not. If so, computer will choose the same standard opening to response to it.

* Answer to question 2:

We plan to build a class named *re\_move* e.g.

*class re\_move{*

*move start;*

*move end;*

*bool capture\_move;*

*Subject captured;*

*};*

Whenever a game starts, the program will create a vector of class *re\_move* called *move\_stack* and using *push\_back* to record every move. When the *undo* command is called, the program use the last *re\_move* in *move\_stack* by vector::back to reversely move the piece from *re\_move::end* to *re\_move::start*, and alsocheck whether it’s a *capture\_move*. If it is, put the captured piece back to the position of *re\_move::end.* Then, *pop\_back* the *move\_stack*. By following these steps, unlimited number of undos can be achieved until the *move\_stack* becoming empty, which is the beginning of the game.

* Answer to question 3:

In order to play four hand of chess the following classes will need to make changes

Grid Class

Expand the grid with an additional 3 rows of 8 cells extending from each side.

The gamerule for whowon() should be changed.

Text Display Class

Expend the text displayed grid the same way as the actual grid to fit a four hand game.

Add two more ways of display pieces for other two players.

GraphicsDisplay class

Expend the text displayed grid the same way as the actual grid to fit a four-hand game.

Add two more ways of display pieces for other two players.

The main function also need to change in order to maintain a round of four players to make their move.

* Final Questions

What lessons did this project teach you about developing software in teams?

1. Teach us how to express my ideas clearly which can be understood by my teammate.
2. We should code different part with low coupling.
3. Learnt how to make a code which be easy used by my teammate.
4. We should make a clear division of labor before starting work.
5. Make module sharing a common purpose.
6. We should start our debug earlier.
7. We learnt how to provide some specific interfaces which can help my teammate understand and use my code.
8. When facing problems, we should calm down first and be patient enough to teammate and our code.

What would you have done differently if you had the chance to start over?

1. We would make the AI become more intelligent, e.g. has some strategy to predict the next few moves.
2. Make our command “undo” better which can undo unlimited moves and can undo some specific move with no segmentation fault.
3. Make a standard opening to help some beginner in chess understand and learn chess better.
4. Polish our graph display to have a beautiful appearance.
5. Organize variables in class better. At this time, we made some useless or duplicate variables because we write the code separately at the beginning.
6. Unified the coordinates of chessboard and pieces in different parts. At the beginning, I set the original point on the left bottom corner; however my teammate set the original point on the left top corner.

* Conclusion

During this cs246 wonderful course, as beginners of C++, we have learnt lots of useful strategies to code our program, which can improve our efficiency and accuracy in coding. We really like this course, although we face lots of problems and difficulties almost during all the term. We think this course is a great challenge for a beginner like us. But we believe it is very helpful. After only one term study, I cannot believe that we can almost successfully finish such a complicated and interesting final project. We really appreciate our instructor and TA who help us a lot. We believe that after making continue efforts, we can learn more and become a good programmer in CS faculty. The final project is very useful for us to have good practices and make a good review for the all term study. We went to full length to finish it. We coded from day to night even in the midnight. We faced some problems when we submit our work. We still have lots of details need to be improved, we will go on working on it after due day.