Chess With a Bot

Kevin Morales-Nguyen
Howie Nguyen
Garvin Ha
Adrian Hernandez Palma
Garrison Astaunda

Chess Masterz

6/1/2020

Team Members and Roles

Kevin Morales-Nguyen: Logic/Intelligence Architect

- Help implement the bot to play against the user.
- Make the Chess pieces and help with the rules of chess.

Howie Nguyen: Logic/Intelligence Architect

- Help implement the bot to play against the user.
- Make the Chess pieces and help with the rules of chess.

Garvin Ha: GUI Development

- Making the graphical aspects and the aesthetics of the chess game
- Consistent interactions i.e the timer will stop for the player that just made their move.

Adrian Hernandez Palma: GUI Development

- Making the graphical aspects and the aesthetics of the chess game.
- Consistent interactions i.e the timer will stop for the player that just made their move.

Garrison Astaunda : Development Operations

- In charge of playtesting and debugging to make sure that newly developed features are implemented smoothly and integrated easily. The project is implemented in a logical manner and all team members are informed about integration specifications and on the same page throughout the project.

Project Summary

The final project is a feasibly functional chess program that allows two players to play chess in an offline setting. Unfortunately, due to a shocking lack of participation and communication from group members, the project was only able to be about more than halfway completed, besides myself working on it for the past couple weeks (Kevin), Howie and Garrison arrived in the final days to give me some feedback and to help patch up some of the bugs and design issues. The foundation for the chess game is solid and clean, with two players the game can be played with minor logical hick-ups but the code was constructed in such a way that it is crash proof and a fair amount of testing was done to ensure it is functional and enjoyable. The project has a very big emphasis on object oriented programming, and I really wanted to use this project as an opportunity to explore the process of design and development. Chess was a great example to use as there are a lot of moving parts to the game and it was our task to make the logical groupings and implement the game and to take it a bit farther to attempt to build a simple bot that would move pieces randomly. Although I did not accomplish building the bot, the program still has a strong foundation and can definitely be finished and used in future, I plan to use it for AI development when I learn more about it in the future. Regardless of the shortcomings I still believe that there were plenty of concepts taught throughout the course that were present in the final release.

Methodology

Topics used in the final project include heavy object oriented programming, classes, abstract classes and abstract functions, inheritance and polymorphism, 2d arrays, arraylists and data structures, a pinch of recursion and threading and a large amount of GUI programming. Object oriented programming was one of the hallmarks of the projects as there are multiple ways to go about abstracting and generalizing a chess game, this was clearly displayed in the implementation. Inheritance and polymorphism were also a perfect fit for the project as a piece could be more specific and although every piece shares the same methods, the way that the pieces behave differs. The player was extended to implement a bot, although the bot was never fully realized it still has a solid foundation that it extends upon, and will definitely be realized in the near future. It goes without saying that a 2d array would be a suitable data structure to represent a chess board, there are declared arraylists for storing information about the game like a player having pieces and hostages and the bot even has a custom structure to store information easily so that when it does crunch numbers it can do so in an organized logical manner. There is a pseudo code segment where I believe threading could be utilized in the bot computations. Finally the GUI component was pretty big as it allowed players to interact with the game, making a big leap from command line programs.

Results

The chess program works quite well and implements a variety of rulecheck systems to ensure continuity with the rules and logic of the game. The main features included are the simple user interface implemented with the processing environment, because the api was so easy to read it made development proceed much better and smoothly then has I developed with java awt, swing or fx. The environment allowed me to easily implement logic and then have it graphically visualized as a program. The rule check system includes moving pieces and checking to make sure that the pieces move according to their respective rules. All of the pieces accomplish this except for the white pawn having the ability to move back one spot diagonally. Another clear design flaw is the fact that pieces can phase-shift through other pieces, it was only at the last minute that I realized this flaw so I was not able to patch it up. In terms of two humans playing the game works very smoothly assuming the players understand the rules of the game. Some other features that were left out are castling and promotions when a pawn reaches the other side of the board. The program keeps track of turns so that a one side cannot make moves twice in a row. Although the program is suitable for player vs player, it still needs a bit of work even for player vs bot, the phase-shift and white-pawn behavior make it unreasonable for a bot to begin training as it may start to accept such moves as normal. Another big feature the game is missing is an end-game check, back to player vs player, the two players could manually detect a check-mate but there is no system in place to automatically detect a check-mate or draw. I underestimated this problem and although I revisited it for about a week it was still a challenging task. The bot was supposed to be a huge part of the program as that is where the majority of the concepts learned later in the class could shine but a delay in scheduling and no one else working on it, I decided to put my effort into producing a reliable program that users could interact with and in that regard I believe the goal was achieved. Other features that were dropped were other GUI features like a start screen, replay option and game statistics, like writing the game history to

a file or displaying the most recents moves, having a timer for timed games. I still believe that all of these thighs could have been accomplished had there been more man power and clear communication and meeting times/dates.

Videos of Project

Live Demo:

https://www.youtube.com/watch?v=o5vaMk9cksg&feature=youtu.be

How to setup and compile code in processing ide or in preferred editor:

https://www.youtube.com/watch?v=zbn2kNeIEl0&feature=youtu.be

Reflections

When looking back on the time I spent on this project I can say with confidence that I improved on my OOP and abstractions skills, it was interesting and fun to break down the game into its components and build it from the ground up, although I was not able to deliver on my promise of a bot I felt that there were multiple instances where I had to think about how to build everything up and use what I learned throughout the course to make wise decisions in terms of the logical groupings and also efficiency. There is definitely a lot that could be improved on, I definitely could have implemented a more diverse set of data structures, a lot of the parts that were dropped included a lot of data processing on collections in able to determine a best possible move or whether a piece's path is blocked. Another aspect of the project that could have drastically been improved upon was the participation and communication of team members, I mean I have no idea what to think when we formed a group and I was the only one working on the actual code, I am shocked and speechless. I mean maybe things hit the fan with covid but this class was already online and the infrastructure was stable so really the work just needed to get done. So I do not know if my group mates just took advantage of the situation or if this is actually their attitude towards group work, but I will not comment anymore on the matter. I suppose next time I should either pick more motivated groupmates or start training myself to do the work of three people. I will definitely be using this codebase for future projects as I learn more about AI, I am very excited to learn more about the subject and have already sparked plenty of thought on how it can relate to chess. The first step would be to finish patching up the rule check system, and then next would be to do the number crunching in the bot class.

Conclusions

_____Beside the fact that I was the overwhelming primary contributor to the project, I still had a blast building it. I did not get it to where I would have liked and I am honestly a bit embarrassed, I said things and did not follow through. But regardless I pushed on and just continued and tried to close out strong, I still believe that the project reached an minimally acceptable state and that it reflects a fair amount of the topics covered in the class, especially OOP. It was interesting to start from a blank canvas and to struggle with it and make something out of it, I think that I have a habit of re-inventing the wheel but I feel like as a student I should be examining the nuts, bolts, and tools very closely so that I can get a better grasp of the bigger picture one day. The project presented is a two player chess program that works with a couple of hick-ups, the bot feature was dropped from the final release and is the reason why a lot of concepts taught later in the class were not covered. The projects cover oop, GUI programming and elementary data structures such as arrays and lists.

Codebase

Files Included:

1. Chess_Game.pde
    - This class serves as the main for the program, it contains a setup() method and a draw() method which sets up the frames and other variables/objects and then updates the frame via the draw() method. The game is actually mostly static so the frame is only updated by a mouse click.
    - Contributors: Kevin
2. Chessboard.pde
    - This class houses the main structure that serves as a chess board, it contains a 2d array of Pieces which serves as a chessboard. The initial pieces are placed onto the board in the constructor and there are methods for gathering information on the state of the board and for debugging.
    - Contributors: Kevin, Howie
3. Move_Engine.pde
    - This class is quite simple as currently it just validates moves that are defined by each piece, but this class would be used extensively with the bot, most of the number crunching would be done by this object.
    - Contributors: Kevin, Garrison,Howie
4. Piece.pde
    - This class is the backbone of the program and is the primary object used throughout the program. The object has member data that specifies its location and team status. The data members held within the piece is used extensively throughout the program to determine movement behavior.
    - Contributors: Kevin, Garrison
5. Player.pde
    - This class is used to represent a player. It's main purpose is to be used to pick up and hold a piece and then to be used as an argument for making a move on the board.
    - Contributors: Kevin, Garrison

6. Bot.pde
    - This class is mainly a stub, it contains primarily pseudo code that shows how the bot might operate and what data it might work with. It shows the possibility of how a thread could be implemented to generate possible moves and validate moves concurrently
    - Contributors: Kevin
7. Chess_Game.java (contains entire program in a single file)
    - This file is generated by the compiler for those that want to port projects to another editor or see how the code works in java.
8. Assets (png's of chessboard and colored pieces)
    - This is not a source file but it is important to note that all of the pictures for the game need to be in the same directory as the pde or exe or else the program will crash.

* I will post a video on how to setup everything in java with processing core.jar and extending PApplet.

*Checkout the myconnect submission for a zip of all the code and assets, the setup video details how to get the pde files to work.

*Video is under Videos of Project page

References

The processing api was a huge help because it provides documentation of the environment and has a boat load of GUI methods that are very easy to use.

https://processing.org/reference/

I used this forum post to figure out how to set up threads in the project, the thread is not actually called so it is basically pseudo-code of how it could be implemented.

https://forum.processing.org/two/discussion/9707/creating-threads-in-processing

There is actually nothing else to reference because in an attempt to hold on to the core value of developing from scratch, I did not look anything else up, watch any videos, or copy and paste any code. Every line of code was written by me and I can provide justification even if it may not exactly be the best answer.

There are no other third party libraries beside the processing environment. Processing is an open source platform and they provide a jar that allows the processing api to be ported from the native processing editor to an editor of choice.