Nick Maggini
Senior Project Paper
CS-480
December 4, 2015

X4

In the amazing game of X4, one can relive their childhood as they drop game pieces into the game board attempting to outwit their opponent and get four in a row. My implementation of this classic game was fun, annoying, frustrating, revealing, exciting, stressful, and above all an amazing learning experience. For the first time in my programming life, I personally came up with an idea, designed, and developed a full working game. Previously it was always small programs, unfinished programs, or projects that were assigned, and not chosen by me. Personally this gave a new light to programming for me. As frustrating and stressful as this project was, for the first time in my life, I had fun programming. This was an amazing revelation for me considering how incredibly close I am to having a programming degree. Before this, programming was always just homework that I just had to do if I wanted to make it through school. Programming just happened to be something that I wasn't terrible at, so I stuck with it. This project taught me that this career field does have potential, not to just be successful, but also have fun doing it.

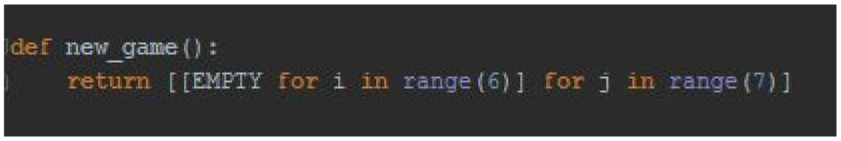
The amount that I have learned from this single project is far more than I ever expected. New coding styles, complicated artificial intelligence, and designing from scratch are just some of the key things I have learned. When I planned this game, I had no idea what was needed to accomplish it. I never have had to do any kind of intensive research about anything. The AI was something that on quick lookup, for Connect Four, was solved. I knew it would not be easy, but when it came to implementing the AI for my game, it was an incredible challenge. I also did not think that modules would be so difficult to work with, or how powerful and helpful they can be for accomplishing tasks.

The artificial intelligence in my program is something that I severely underestimated. I did not take into account for how exponential an AI can become. In Connect Four, there are 4.53 trillion board combinations and if 100% perfect AI is desired, good luck checking all of them for the win. It took far more research than I expected, but during the research I learned a tremendous amount. Previously I had no knowledge of AI, and after this project I truly respect anyone that works with AI. The things that AI can do are amazing. A single algorithm can be applied to so many different things. Some examples that the algorithm A* could do are video game winning conditions, fixing spelling errors, or finding the fastest way to a destination. A GPS could make great use of A* by finding the fastest way to a location while easily avoiding traffic backups.

The A* algorithm is really impressive because it is given a start state and an end, and these points can be anything. In my case it is the current board state and winning a game. The algorithm begins at the start point and finds the shortest path to the end goal. A* is great for pathfinding just about anything. I found a really interesting implementation of it for finding the shortest path to an object in a video game, even if other obstacles are in the way. A* works by generating a list of all the possible steps toward the goal from the current position. It stores them in a priority queue based on the distance towards the goal with the closest being first. Then recursively select the closest path until the goal is reached. All of the paths are stored because if a path fails, for instance in my game, when a winning move gets blocked, A* can pick back up on the next best path.

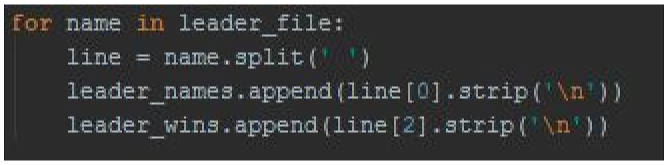
One of the more interesting things that I learned about AI is that a single algorithm does not have to be the only thing trying to solve a problem. To make amazing AI's many different methods and algorithms can be combined to make an ultimate AI. The problem that the AI is trying to solve can drastically change what algorithms, and what methods should be used. In a game of Connect Four it turns out that an algorithm called Minimax is one of the best to implement an AI. This is something I didn't know about, and definitely should have been on my rubric. Not only that, but the depth-first search that was on my rubric should be implemented through minimax. Then for it to be any good, without taking forever for the computer to think, alpha-beta pruning should be added. This removes any branches that waste the computer's time checking. This pruning can be a lot of different things, removing any losing games is obviously the first step for a game like X4. Removing any draws is a decision that is needed to be made on whether one is looking for the AI to be great or perfect. If they are removed the computer must win, or it could lose. This is because in Connect Four if both players play perfectly, most games can end in a draw. If all the draws are removed from the search tree, and the opponent plays perfectly and/or goes first, the AI could fail. More methods can also be used to perfectly fine tune an AI.

I also learned a lot about this impressive new language to me, Python. This language is amazing, and has so much power behind it. This project gave me a ton of experience with the language, and I have grown to love Python. One of my favorite feature of this beautiful language is that out of the pages and pages of code in my project, I can not seem to find a single semicolon. My pinky greatly thanked me for it. For those who care about more than syntax, Python offers amazing things like list comprehension(ex. 1) and iterating over objects(ex. 2) easily. List comprehension is a great way to initialize a list with a value. Iterating over objects allows one to easily loop without worrying about off by one errors, it loops for each object in a list no matter the length.

1. 

```
def new_game():
    return [[EMPTY for i in range(6)] for j in range(7)]
```

(List Comprehension)

2. 

```
for name in leader_file:
    line = name.split(' ')
    leader_names.append(line[0].strip('\n'))
    leader_wins.append(line[2].strip('\n'))
```

(Iterating)

Starting my project was a tremendous challenge itself. Finding a decent IDE to program in was a priority of mine. If I was going to spend the next few months programing, there was no way it was happening in VI or Nano. Out of a recommendation I looked up PyCharm. After using it, yet again I found something new that I fell in love with. PyCharm is so helpful, clean, and easy to use. None of the confusing menus of Visual Studios and how I can't find how to change a feature or even text color. The biggest reason I would recommend PyCharm is because of its amazing code analyzer. Being new to Python, it recommended exactly how to format my code to be 'pythonic'. Even with simple syntax that I did not encounter before, PyCharm was helpful with suggested changes to fix simple problems that could have had me stuck for way too long. It even fixed my ridiculous spelling errors after programming way to late into the night. If I ever program in Python it will always be using this IDE, it is the best I have ever used.

When beginning this project I had to decide how to make a game in a language that I had yet to learn. After some research I found this seemingly amazing pygame module. Essentially pygame is SDL for Python, helping grab events and displaying graphics. The documentation decent enough to help me learn to use it. What it was missing was how to get it working. Everything I tried, everything I looked up, just did not work. I spent a gross amount of time just trying to import pygame into my unstarted program. I should've found a better way to make my game, but I had invested too much time, I was dug in, and I didn't want to change. My persistence did pay off, and in the end I am glad I stuck with it. My problems were mainly having to do with Python and everyone in the Python community being a lot like me. People get dug in, and just don't want to change, and this is why there are so many versions of Python. People just do not want to update their old code when new versions come out. So what I learned is that modules not only are not the easiest thing to install, but they need to be the same version of Python. In my case not just Python 2 versus 3. I needed an exact version, when I finally got everything working I had Python 3.2.5 and a version of pygame that I could import.

Now that I had a working pygame and version of Python, I began making the game after some basic design. Later I will extremely regret not designing more. Once I had a finished game and working AI, I set out to get some of the networking stuff done. This is where I looked for a socket library to help speed my project along and meet the project's deadline. Nothing I could find worked. I must of found, installed, and gave up on five different socket libraries because they just did not work with my version of Python that I already wrote a game for. The amount of time that I worked and wasted because of Python version compatibility makes me regret so much.

Once I finally found a working library for networking I programed a way to get a chat system working with the help of a tutorial. It was awesome, I wrote a little program to instant message anyone on the my site. I was even able to message to my friend down state. When it came to the networking the game, I hit a brick wall at 100mph. I messed up, I didn't know what to do. I tried everything, but I could not find a way to get my game to display on my web server based on how it was built. Not designing my entire program from the beginning was my biggest downfall of this project. Being stuck for far too long, I did the best thing that I could think of, I put it on the back shelf for a later time and focused on AI and making a better game.

If I could start this project from the beginning the biggest thing that I would do differently is plan better. Starting with my proposal I set myself up for a really rough road. I didn't really take it seriously enough, I put some points on there that I did want to do, but didn't really think if I was capable of it. Mainly I don't understand why I thought it was a good idea to focus on AI because of my weak knowledge of math. Not being a computer science major and being a networking computing major seriously made this task extremely difficult. On top of that once the committee finally approved my proposal they warned me that this project was going to be difficult. Yet for some reason I ignored the group of people with PhD's. If I hold onto anything that I learned from this project, I really hope I never ignore the advice of the people that are smarter than me. Something my Mom always told me, but of course I can be stubborn.

The things that I would do differently when it comes to the actual project is focus on things that I know more about, and design. I did learn an incredible amount about AI and networking, but something that will help me with my whole career is how important it is to have a well thought out, and decent design of the project that I am about to attempt. I have never had to do anything on this scale before, and I didn't put enough thought into the design. This is a huge mistake that I will always try to avoid on any future programs.

My project is organized in way that I thought was most logical to me. Whether or not this is the correct choice I am undecided. If I were to reorganize, though, I may break things down a little smaller than I did just for understandability. Currently it is organized by being broken down into files of: GUI, Game, Login, Chat, Leaderboard, and each AI type files. There is also a database of every turn eight board combination possible, and what the outcome of the game should be for player one. This is useful for checking and speeding up the unbeatable AI.

The hardest part of this project was definitely trying to get a perfect and unbeatable AI working. I put more time into this than any part other part of the project by far. It became really important to me, and weeks of research and trying different methods all resulted in failure. One day when I was about to give up, I finally found John Tromp's research (Lichman). He compiled a database of every turn eight board combination, and whether player one should win, lose, or draw. His data took over 40,000 computation hours to finish with 67,557 board combinations. This gave me great hope, I could see the light at the end of the tunnel.

I took the data and found a way to get it to work with my project. I could, in less than five seconds, take all his data and push it into game boards for my project. I can easily look up any of these boards based on my current board state to see if I should win or not. The problem that still existed though is all of the 67,557 boards were for turn eight only. At this point my AI could only look two turns ahead without taking too long to compute. So the AI could see if it should win on turn eight if it plays perfectly from then on. It also needed to get to one of these board states without putting itself in a losing position before turn six, because of only being able to look two turns ahead. Then if the actual win was not within two turns after turn eight, it needed to play absolutely perfectly for potentially another 31 moves. The perfect unbeatable AI was only perfect some of the time, which isn't really perfect at all. I was stumped again.

After even more research into the unbeatable AI helped me find a little more insight, but everything I found was just repeating what I already knew. After awhile, I really thought about it, and the only people I could find that actually completed this task were people that devoted their lives to AI or mathematics. There were a total of three people that had proof of an unbeatable AI that I found. These people were so far beyond my intelligence in the subject matter what little information they had about accomplishing a perfect AI was not much help. I decided that I picked a task that I was just not able to finish in any decent amount of time. I chose instead to just try and make the best AI I could and hope it just never has to play against a professional Connect Four player.

The project in whole was an amazing experience. Creating the game took me awhile but I actually enjoyed it a lot. Making a working video game from nothing all on my own is an amazing feeling. I learned a ton about what goes into making games. A lot more infinite loops than I thought. While the game is running the computer is almost always in some loop just waiting for events. Another feature that I enjoyed creating was the leaderboard. I may have come up with a weird way of making it work, but it is mine, it works, and I like it. It was a feature that was added after the game, menu, and GUI was already done so it took a few work arounds. Sometimes though working with what I am given and making something work can be extremely gratifying.

The project became way more research than I ever imagined. I did not mind the research at all though. When I was learning about something that actually matters to me things just come easy. Implementing the research into my project is a slightly different story, but that's only because of the frustration of my inability to get something working that I just spent forever trying to learn about.

X4 was a project that I put a incredible amount of effort into even if it was difficult for me. I learned so much while working on this project that I will continue to use throughout my programming life. Listening to the committee on how this project was going to be hard is something I really wish I did. My unbeatable AI is something that maybe needed a little less effort to allow for more focus on other things. Now that I have had time to work on my project, I would have researched more about Connect Four, AI’s, and the networking involved, so I could have make a more appropriate grading rubric. The rubric, as is, may not agree with it, but I truly believe I deserve a decent grade reflective of all of my hard work and knowledge gained. This project is something I put a lot of time into, and for the longest time I couldn’t say it, but now, I finally am proud of a program that I made.

Sources and Links

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

List Comprehension:
http://stackoverflow.com/questions/2397141/how-to-initialize-a-two-dimensional-array-in-python

Pygame:
http://pygame.org/hifi.html

PyCharm:
https://www.jetbrains.com/pycharm/

Tornado:
http://www.tornadoweb.org/en/stable/