Project Proposal
Name: Vishnu Pathmanaban
AndrewID: vpathman
Section: M

OUTLINE TAKEN FROM: https://www.cs.cmu.edu/~112/notes/term-project.html#tp1

Project Description [5 pts]: The name of the term project and a short description of what it will be.

The name of my term project is "ChessSensei". ChessSensei will be a chess software package with two modes: player versus player and player versus AI. At the user's request, the ChessSensei AI will give the player feedback on better moves they could have made.

Competitive Analysis [5 pts]: A 1-2 paragraph analysis of similar projects you've seen online, and how your project will be similar or different to those.

ChessSensei falls under the category of chess software packages. The goal of the software is to simulate a chess match between two players or against an AI, and during these matches the software will give the player feedback on better moves they could have made. Some similar chess software packages are Chess Master by Steve Liu, lichess.org, and chess.com. Similar to these other chess software packages online, ChessSensei will have two modes: player versus player and player versus AI. Additionally, ChessSensei AI will operate on a minimax algorithm with alpha-beta pruning, similar to Chess Master by Steve Liu. However, ChessSensei will differ from these projects in its usage of its AI algorithm. Instead of just using its AI algorithm as an opponent, ChessSensei will also use its AI algorithm to give the player feedback on better moves they could have made. I will need to concentrate on my ChessSensei AI feedback system to be competitive.

Dimension 1	Accurate Representation of Chess
Dimension 2	Artificial Intelligence Level
Dimension 3	Adaptive Move Feedback
Dimension 4	Online Multiplayer
Dimension 5	Scalable GUI

	Dimension 1	**Dimension 2**	**Dimension 3**	**Dimension 4**	**Dimension 5**
Chess Master	Yes	Basic	No	Yes	No
lichess.org	Yes	Advanced	No	Yes	Yes
chess.com	Yes	Advanced	No	Yes	Yes
ChessSensei	Yes	Basic	Yes	No	Yes

Structural Plan [5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or objects.

First, I will create a Piece superclass with specific chess piece subclasses describing move legality. Next, I will create a Board class that acts as a two-dimensional data structure containing and displaying all the chess pieces. Then, I will create a broad ArtificialIntelligence function that calls upon the four smaller functions: the tree generation algorithm, the board evaluation algorithm, the minimax algorithm, and the alpha-beta pruning algorithm. Finally, I will use Carnegie Mellon University's "MyModalApp" framework for mode control between the following modes: the start menu, player versus player, and player versus AI. In both game modes there will be a keyboard function to enable ChessSensei AI feedback. As of now, all of this code will be well organized within one file. However, if necessary I will split the code into individual files that will be imported in a main executable and called upon.

Algorithmic Plan [5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project.

The trickiest part of the project will be creating and implementing the ChessSensei AI. The ChessSensei AI will be comprised of four key algorithms: tree generation, board evaluation, minimax, and alpha-beta pruning. The tree generation algorithm will generate the possible future board positions based on the possible moves pieces can make. The board evaluation algorithm will assign quantitative values to each piece to compute the overall board value of each move along the aforementioned tree generation. The minimax algorithm will calculate the best move from the aforementioned tree of moves using the board values. During the AI's turn, the minimax algorithm will prioritize maximizing board value, and during the AI's opponent's turn, the minimax algorithm will prioritize minimizing board value. With this logic, the minimax algorithm will select the tree path that most effectively benefits the AI and restricts the AI's opponent. The alpha-beta pruning algorithm will optimize the AI by ruling out tree paths that the AI nor its opponent will ever take. Alpha will be the lowest value the AI is willing to accept for itself, and beta will be the highest value the AI is willing to accept for its opponent. Using these bounds, the alpha-beta pruning algorithm will reduce the space of the tree search.

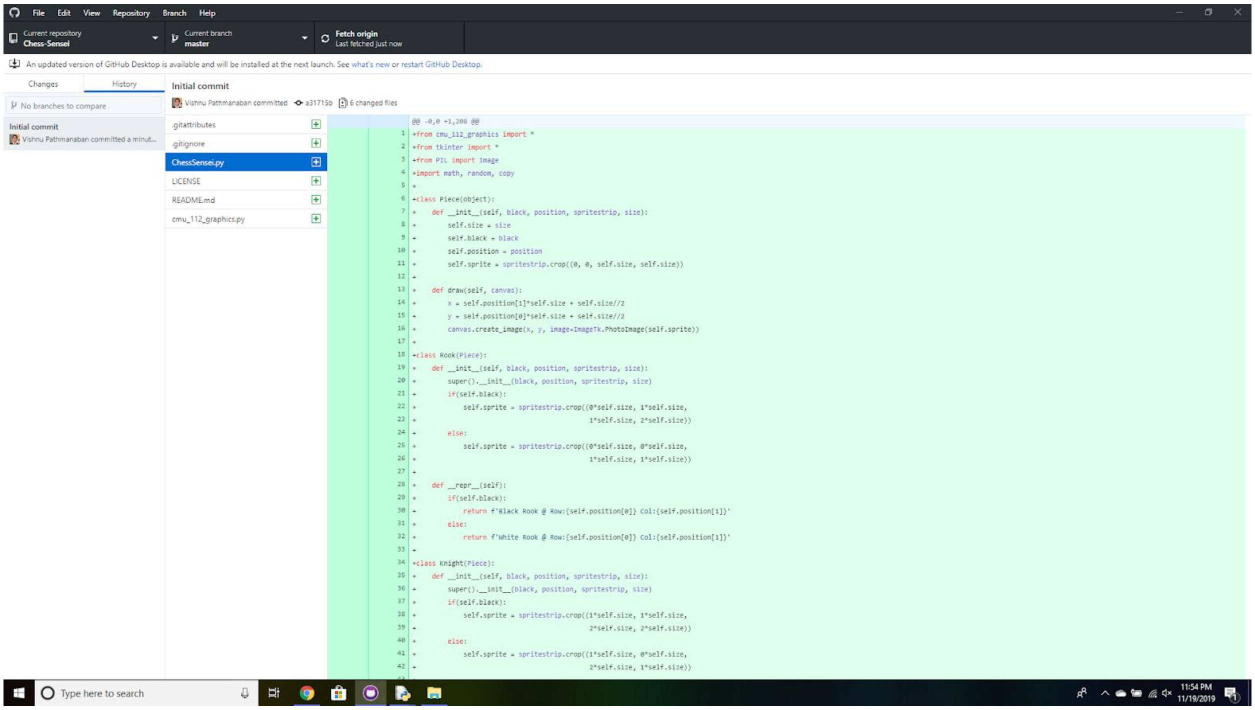
Timeline Plan [5 pts]: A timeline for when you intend to complete the major features of the project.

Feature	**Date**
Basic chess board with scalable GUI	11/20/19
Advanced chess board with move legality	11/22/19
Tree generation and board evaluation	11/24/19
Minimax and alpha-beta pruning	11/26/19
Implementation of feedback option	12/3/19
Final touches and submission	12/4/19

Version Control Plan [3 pts]: A short description and image demonstrating how you are using version control to back up your code. Notes:

- **You must back up your code somehow!!!**
- **Your backups must not be on your computer** (ideally, store them in the cloud)

I plan on using version control to back up my code with Google Drive and GitHub servers. I plan on backing up my code to the cloud using GitHub Desktop and Google Drive. GitHub Desktop will store my code on my GitHub account, and Google Drive will store my code on my Andrew account. With this method of version control I will be able to access previous versions of files and old builds of my code as well.



An updated version of GitHub Desktop is available and will be installed at the next launch. See what's new or restart GitHub Desktop.

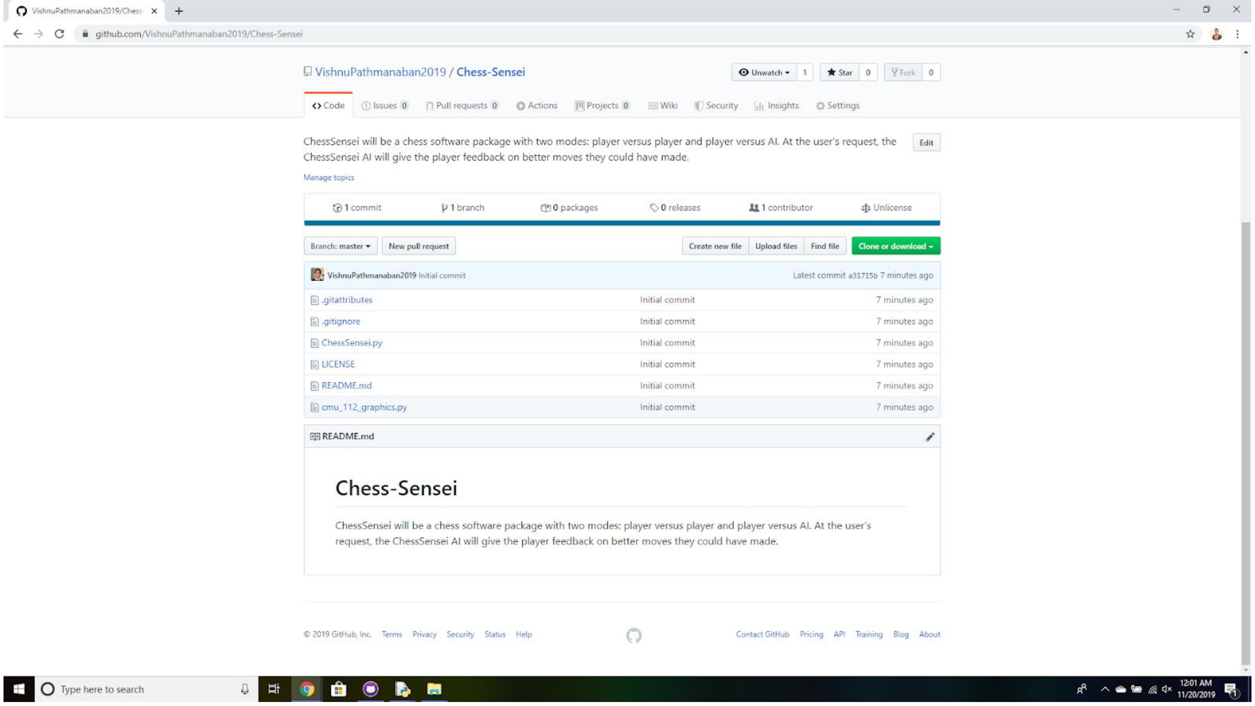
Initial commit
Vishnu Pathmanaban committed a minut...

Initial commit
Vishnu Pathmanaban committed a31715b 6 changed files

.gitattributes
.gitignore
ChessSensei.py
LICENSE
README.md
cmu_112_graphics.py

```
@@ -0,0 +1,208 @@
+from cmu_112_graphics import *
+from tkinter import *
+from PIL import Image
+import math, random, copy
+
+class Piece(object):
+    def __init__(self, black, position, spritestrip, size):
+        self.size = size
+        self.black = black
+        self.position = position
+        self.sprite = spritestrip.crop((0, 0, self.size, self.size))
+
+    def draw(self, canvas):
+        x = self.position[1]*self.size + self.size//2
+        y = self.position[0]*self.size + self.size//2
+        canvas.create_image(x, y, image=ImageTk.PhotoImage(self.sprite))
+
+class Rook(Piece):
+    def __init__(self, black, position, spritestrip, size):
+        super().__init__(black, position, spritestrip, size)
+        if(self.black):
+            self.sprite = spritestrip.crop((0*self.size, 1*self.size,
+                                            1*self.size, 2*self.size))
+        else:
+            self.sprite = spritestrip.crop((0*self.size, 0*self.size,
+                                            1*self.size, 1*self.size))
+
+    def __repr__(self):
+        if(self.black):
+            return f'Black Rook @ Row:{self.position[0]} Col:{self.position[1]}'
+        else:
+            return f'White Rook @ Row:{self.position[0]} Col:{self.position[1]}'
+
+class Knight(Piece):
+    def __init__(self, black, position, spritestrip, size):
+        super().__init__(black, position, spritestrip, size)
+        if(self.black):
+            self.sprite = spritestrip.crop((1*self.size, 1*self.size,
+                                            2*self.size, 2*self.size))
+        else:
+            self.sprite = spritestrip.crop((1*self.size, 0*self.size,
+                                            2*self.size, 1*self.size))
```

GitHub Desktop



VishnuPathmanaban2019 / Chess-Sensei

ChessSensei will be a chess software package with two modes: player versus player and player versus AI. At the user's request, the ChessSensei AI will give the player feedback on better moves they could have made.

1 commit · 1 branch · 0 packages · 0 releases · 1 contributor · Unlicense

VishnuPathmanaban2019 Initial commit — Latest commit a31715b 7 minutes ago

.gitattributes	Initial commit	7 minutes ago
.gitignore	Initial commit	7 minutes ago
ChessSensei.py	Initial commit	7 minutes ago
LICENSE	Initial commit	7 minutes ago
README.md	Initial commit	7 minutes ago
cmu_112_graphics.py	Initial commit	7 minutes ago

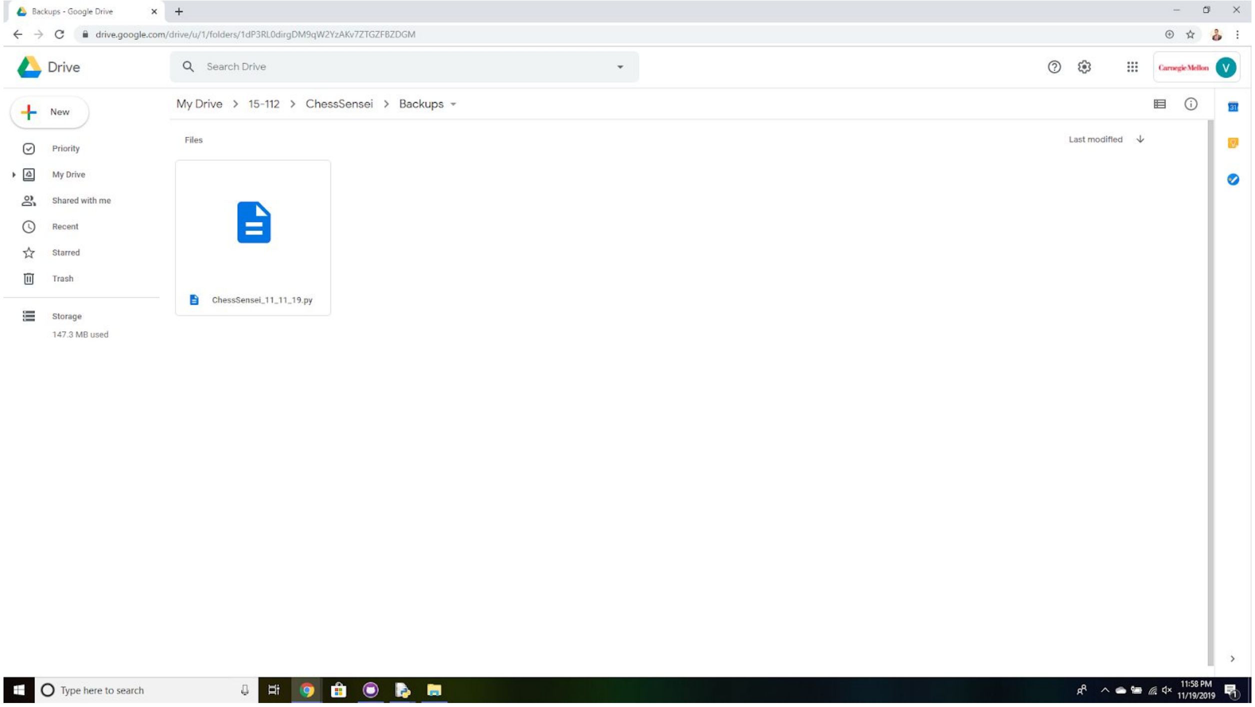
README.md

Chess-Sensei

ChessSensei will be a chess software package with two modes: player versus player and player versus AI. At the user's request, the ChessSensei AI will give the player feedback on better moves they could have made.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

GitHub Account



Google Drive

Module List [2 pts]: A list of all external modules/hardware/technologies you are planning to use in your project. Note that any such modules must be approved by a tech demo. If you are not planning to use any additional modules, that's okay, just say so!

I am not planning on using any external modules in my project besides tkinter and pillow.

TP2 Update [11/26/19]:

No major changes to my overall project proposal. Some minor changes include:

- ChessSensei will give advice on moves the player SHOULD make (Project Description)
- ArtificialIntelligence function renamed pruningMinimax (Structural Plan)
- Plan on separating code into three files: Pieces, Board, and Main (Structural Plan)
- AI will return explanation for advice by showing opponent's threats (Algorithmic Plan)

TP3 Update [12/05/19]:

OVERALL KEY FEATURES:
1. Menu user interface
2. Chess user interface

3. Highlights legal chess moves for beginners
4. Player versus player mode
5. Check and checkmate detection
6. Player versus custom AI mode
7. AI difficulty
8. Advice function
9. Advice explanation display
10. Scalable GUI for different devices

- Added advice explanation that works with chess board locations
- Added difficulty options for chess AI
- Added checkmate detection and crash avoidance
- Customized minimax algorithm to prioritize checkmates
- Added checkmate specific advice to advice function
- Moved console text to GUI display under chess board