CS 15112 Term Project TP 1

Hannah Ling

Project Description

Codenames 112

My project is a digital version of the game Codenames by Czech Games Edition. In it, two teams--red and blue--race to find their team's 8 words on a 5 by 5 grid of cards. Their respective codemaster, who can see the "answer map", gives them a series of clues, each consisting of one word and one number.

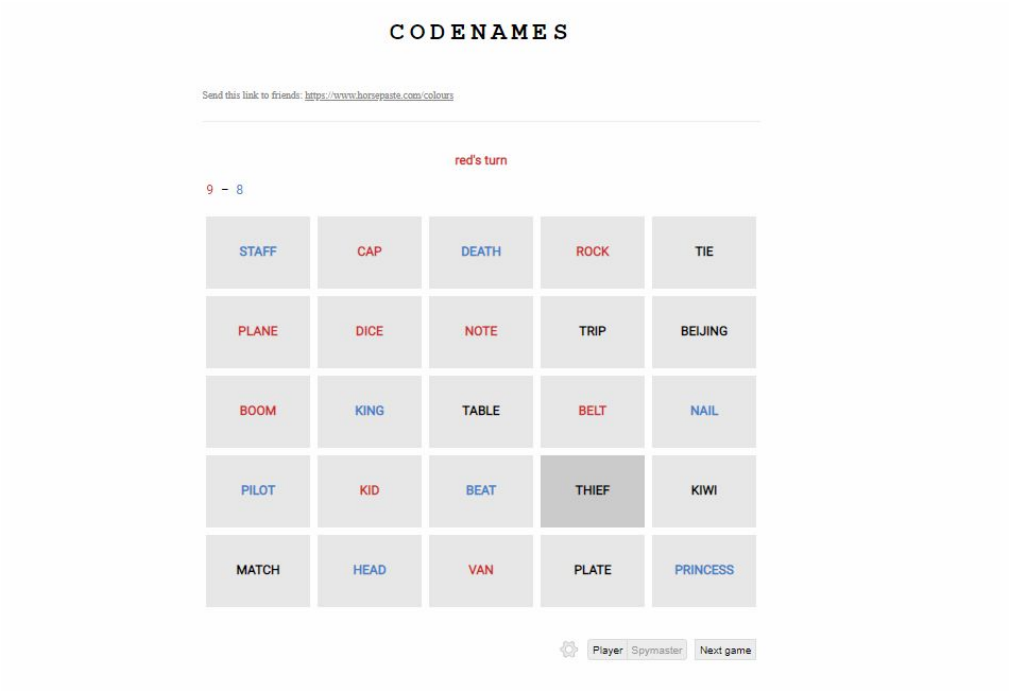
I want to recreate the game using TKinter--draw the board of words, have the codemaster input the clues, then have the teams guess by clicking on the board, which will update to show the results. The purpose of creating this project would be to have a group be able to play even when they don't physically have the game, and to allow them to input their own custom boards and lists of words.

I also wanted to add a singleplayer "training mode", which has the computer using Natural Language Processing(NLP) to come up with clues that the user has to guess. If there's time after completing the main body of the project, I wanted to explore a multiplayer mode as a challenging extra bonus step.

Competitive Analysis

Other online version of Codenames

.Through a Google search, I found one online version of codenames that already exists.



Some of the pros of this competitor include the sleek design and aesthetic appeal, and the fact that it is multiplayer--there is a game code you can input to join the same game as your friends.

Ways I want my project to be similar:

Minimalistic design

Click cards to choose them

Toggle back and forth between Spymaster and Player mode

Displays who's turn it is and current score

Ways my project will improve on it:

Option to play against a computer player in a "training mode" -- computer gives clues and you guess

Timer imposes time limit for giving clues and guessing like the original game

Codemaster can type in their clue

Support to input a custom list of words

More user friendly/easier to understand in general

Structural Plan

Files

The majority of the code will be in one graphics/animation file, `codenames.py.`

Modes (screens):

Start screen

Play Against Computer Screen

Player screen

Codemaster screen

Game Over screen

Objects:

Red/Blue turn: *Boolean flag to tell whose turn it is. First turn is random.*

Game Over flag: *Boolean flag to tell if the game is over*

Word Bank: *List of random words that users can add to*

Board: *2D List of Words from word bank*

Word: *Custom object that has a value, color "allegiance", and isClickedOn variable*

Map: *2D List of Words with color "allegiance" visible. Only visible in Codemaster mode*

Timer: *Integer that starts at 90 seconds and counts down. Resets every time turn changes*

Guesses left: *Integer that starts at # of guesses given by the current clue and decreases every click*

Score (red version and blue version): *+1 for every correct word guessed, first to 9 wins(8 if they went first)*

Current Clue: *Tuple that contains a clue(string) and guess # (int). Will check to see if input correctly.*

Clue list(red version and blue version): *List of previously given clues*

Functions needed:

```
timerFired()
run()
init()
keyPressed()
mousePressed()
redrawAll()  # drawBoard(), drawScore(), drawCluesList()
clueInput(clue)#User input, need to do research in TKinter for how to do this
inputWords(wordList) #User generated word list
randomWords() #Generate list of 25 random words from word bank or user input
randomMap() #generate map configuration
loadWords() # place words into map in random order, update word “allegiance”
depending on what the map says
```

Algorithmic Plan

Natural Language Processing

Current plan is to use NLTK, a NLP library in Python to find a list of synonyms to come up with the clues. I also want to explore using similarity to come up with clues for more than one word at a time.

To find a clue from two words, make the first word into a Synset, and then look through the synset’s hypernyms and hyponyms, calculating which of them is most similar to the second word. Then return that.

Not the best algorithm, but the simplest I can think of for now. A more advanced version would involve searching through bodies of text like the Brown Corpus of words to find words that appear in a similar context rather than just the hypernyms and hyponyms.

Check out the nlpdemo.py file!

Loading User-Inputted Words into Word Bank

```
**Use TKinter Entry to get it as a string**
For word in inputString.split(newline)
     wordBank.add(word)
```

Loading the Board

```
def randomWords():
     twentyFiveWords = []

     for i in range(25):
```

```
            index = random.randint(0, len(wordBank))
            randWord = wordBank[index]
            randWord = Word(randWord, “neutral”, False)
            twentyFiveWords.append(randWord)
      return twentyFiveWords
```

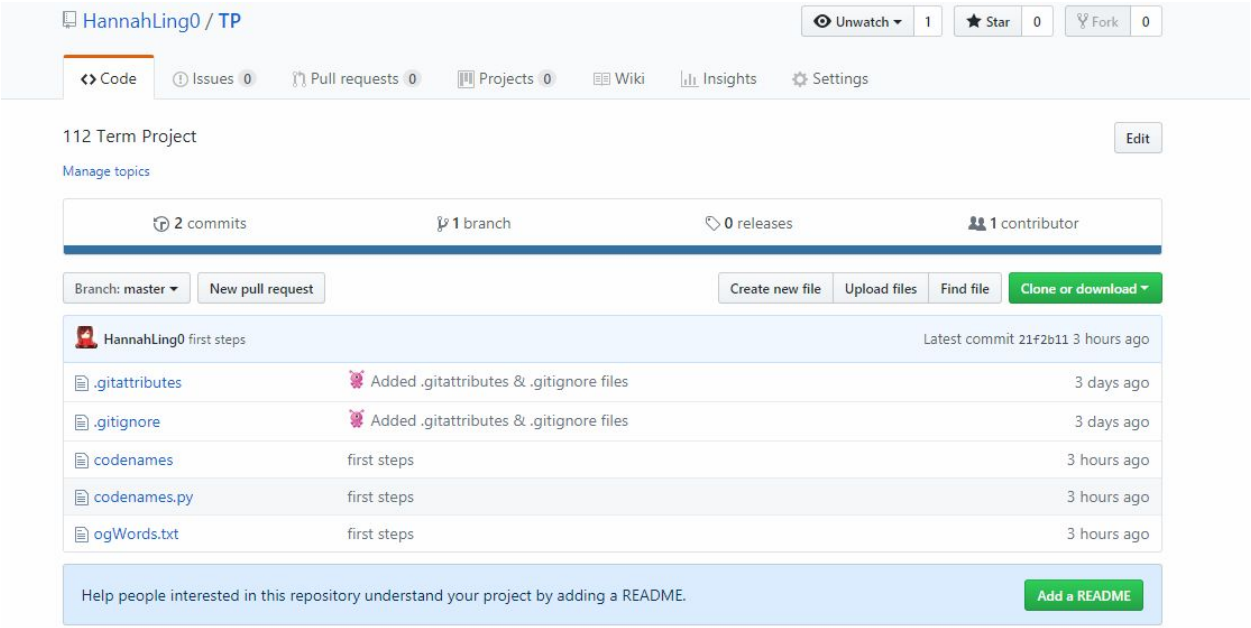
```
def randomMap():
      map = []
allegiances = ["red"] * 9 + ["blue"] * 8 + ["neutral"] * 7 + ["assassin"] *
1
For i in range(25):
      item = allegiances[random.randint(0, len(allegiances)]
      map.append(item)
      allegiances.remove(item)
```

```
def loadWords(map):
      for i in range(len(map)):
            thisWord = words[i]
            thisWord.allegiance = map[i]
            map[i] = thisWord
```

Timeline

Phase	Feature	Date
TP1	Basic board and random words	11/20
TP2	Timing and score	11/23
TP2	Allow user to input custom board	11/28
TP2	Computer generated clues using NLTK	11/28
TP3	Fix aesthetics	12/6

Version Control



I plan to use a Github repository for version control.

Module List

- NLTK - Natural Language Processing
- Sockets - multiplayer (AFTER MVP)