Connect 41

Bouillette Nicolas
Member
Fantastic Four
Garges-les-Gonesse, France
bouillette.nicolas@outlook.fr

Ren Luca
Member
Fantastic Four
Aubervilliers, France
renluca.pro@gmail.com

Robidou Lucas
Member
Fantastic Four
Lanester, France
luc.robidou@gmail.com

Schirinzi Francesco
Member
Fantastic Four
Geroldswil, Switzerland
schirfr1@student.zhaw.ch

***Abstract*—During the second semester at Hanyang University, we were asked to make a project that involved both Software Engineering and Artificial Intelligence. Our team decided to make a game, and we chose to implement "4 wins" (also known as "Connect Four"). This game is a turn-based game, which involves two players trying to connect four of their checkers in a row while preventing their opponent (the other player) from doing the same. This game will be playable through a web interface, and the outcomes will be saved on the server. Thus, we will be able to feed the AI with every game played before. Players will then be able to face the AI in the game.**

TABLE I
ROLE ASSIGNMENTS

Roles	**Name**	**Task description**
User	Robidou Lucas	To express its need and to justify it To use the software (in order to test it) To criticize what is made by the team
Customer	Schirinzi Francesco	To make customers happy To get what I asked for To talk we the Developer team and negotiate
Software developer	Bouillette Nicolas	Producing clean, efficient code based on specifications Testing and deploying programs and systems Fixing and improving existing software
Development manager	Ren Luca	To deliver result to the Customer To set clear goals for the development team To provide guidance and make sure the timeline is followed To evaluates and ensures the completion of tasks and projects

I. Introduction

Everyone had this experience in their lifetime. Either you had a bet with your brother to get the bigger room or to win a bet against a friend. Normally these bets are fought with games. Card games, Chess or other games like Connect Four. In life sometimes it's crucial to win such a bet. But stay close, you don't have to worry anymore about losing important games with our Software. Using new technologies such as Artificial Intelligence we create an app that gives you at any moment the winning steps. Connect 4I does not only know long-known moves but also finds new moves as the software is trained with each game. To reach this goal, we use already established algorithms and make them better. We will not only provide you with a tool to win a bet but also make it to you possible to reach your desires and dreams.

II. Requirements

A. *Functional requirements*

- As a system, I have to display an error message when I detect an error in order to warn the user.
- As a system, 5 seconds after displaying an error message I have to shut down the system in order to preserve it.
- As a system, if the player did not place a pawn during the indicated time I must have played in his place in order to not block the game.
- As a system, as soon as I play in the place of a player, I have to play randomly in order to be fair.
- As a system, once the game is over I have to suggest to the user if he wants to restart or leave the game in order to not interfere with his gaming experience.
- As an AI, I have to choose a shot in less than 3s in order to decrease the user's waiting time.
- As an AI, I have to play the best possible shot according to my degree of difficulty in order to entertain the user.
- As an AI, I have to counter all the opponent's shots that would be winners in order to entertain the user.
- As a user, I can click on "PVP" button in order to play against a human.
- As a user, I can click on the "AI" button in order to play against a machine.
- As a user, I can click on the "Options" button in order to adjust the AI level.

- As a user, I can click on the “Help” button in order to have the help manual.
- As a user, I can click on the “Quit” button in order to quit the application.
- As a user, after choosing the “Options” button, I can click on the “Easy” button in order to have a easy gaming experience.
- As a user, after choosing the “Options” button, I can click on the “Normal” button in order to have a normal gaming experience.
- As a user, after choosing the “Options” button, I can click on the “Hard” button in order to have a hard gaming experience.
- As a user, during a party, I can click on the “Retry” button in order to retry the game.
- As a user, during a party, I can click on the “Stop” button in order to stop the game.
- As a user, after choosing the “Retry” / “Stop” / “Quit” button, I can click on the “Yes” button in order to validate my choice.
- As a user, after choosing the “Retry” / “Stop” / “Quit” button, I can click on the “No” button in order to cancel my choice.
- As a user, during a party, I can go back once in order to revert the position of my token.
- As a user, at the end of a game, I can click on the “Again” button in order to extend my gaming experience.
- As a user, at the end of a game, I can click on the “Menu” button in order to return to the main menu.

B. *Non-functional requirements*

- The application must have a graphical interface.
- The application must start in less than 5 sec.
- The application must be developed within a limited cost.
- The application must have a maximum high definition.
- The application must use a minimum hardware requirement that is relevant to this game.
- The application must be designed in an efficient manner.
- The application must allow playing in less than 3 clicks.
- The application must be ready for December 15th.
- The application must comply with ISO and IEEE standards.
- The application or parts of it have to be developed in an object-oriented language.

III. Development environment

A. *Choice of software development platform*

For the platform, we will be using Linux OS because we’re accustomed to it and it’s a open source OS, it is also easier to install certain programming software on it.

We plan to use Javascript for the front end (user interface) eventually with P5.js to draw the grid more easily. The back-end part will be done in Python3 with scikit for the machine learning part. Indeed, scikit is an efficient tool for doing data classification (winning moves / not optimal moves for exemple) and is used by a large community, which is reassuring (it means that a lots of users may have already encountered and solved most common issues).

To resume, we will be using Javascript 1.8.5, python3.7, sklearn (scikit) version 0.21.3. Our OS are mainly debian 10 and ubuntu 18.04.
We chose those version because they’re either the latest stable version or because we are accustomed to it.

Apart from an (optional) server, our project is totally free of charge, since most of our tools are free.

B. *Software in use*

Many different approaches have been tried to obtain the best Connect 4 AI player.
One of the early researched approaches was made by Victor Allis in his master thesis "*A knowledge-based Approach of Connect-Four*"(1). Here Allis investigated a Shannon C-type strategy, which is based on human strategies and good "rule of thumps" for winning Connect 4. This approach is not the most efficient or most accurate approach, but it is easily understandable and gives some fairly good results.
Though at the time computation time was a big limitation on the solution.
An approach you can find in multiple different papers are the minimax and alpha-beta heuristic algorithm e.g. "*Heuristics in the game of Connect-K*" by Jenny Lam(2).

IV. Specifications

A. *Requirements*

- The system wait 30 seconds for the player for each round, if there is no response, the computer will play randomly for the player.
```
if noResponse within 30 seconds:
        ai.playForPlayer()
else:
    player.play()
```
- When the game end, a pop up will appear with the choice to restart the game or quit to the main menu.
```
ask player “play again?”
if player.answer is yes:
  restart
else:
  quit
```
- The AI will check his difficulty and use the function designated to play as the level chosen by the player
```
in cases difficulty is
        easy then ai.load_easy()
        normal then ai.load_normal()
        hard then ai.load_hard()
ai.startToPlay()
```
- The system will check if there is a winning move for the player, if there is, the computer will play to prevent it.
```
if player.CanWinNextTurn():
  ai.counterUser()
```

```
else:
    ai.play()
```

- The player can click on the QUIT button to go back to the main menu.

```
If player.quit()
    game.exit()
    game.showMainMenu()
```

- The player can click on the Option button and a pop up will appear where he can chose the difficulty setting of the AI.

```
if player.clickOption()
    game.showChoiceDifficulty()
```

- The player can click on the Retry button and the system will reset the board.

```
if player.retry():
    game.exit()
    game.start()
```

- The player can click on the Stop button to stop the timer for the autoplay until he click on it again to play.

```
if player.stopAutoplay()
    game.timer.stop()
```

- The player can click on the go back button to revert the play he and the computer made for the previous turn.

```
if player.resetTurn()
    ai.cancelLastMove()
    player.cancelLastMove()
```

References

[1] Victor Allis, "A knowledge-based Approach of Connect-Four", 1988, http://www.informatik.uni-trier.de/ fernau/DSL0607/Masterthesis-Viergewinnt.pdf

[2] 2: Jenny Lam, "Heuristics in the game of Connect-K", year?, https://pdfs.semanticscholar.org/35de/5f75444e1fc1c96cb6378393f3ff24a74099.pdf

[3] 3: https://medium.com/@ODSC/how-300-matchboxes-learned-to-play-tic-tac-toe-using-men