Implementation of Tic Tac Toe Game Using PyGame

Team Members:

Darshan Sandesh K S	[1VE20CS035]
Bhaskar P	[1VE20CS024]
Pranay G	[1VE20CS040]
Deepak J	[1VE20CS037]
Sanath Kumar	[1VE20CS025]

Guide: Shilpa Hariraj

OUTLINE

Abstract

The Tic Tac Toe game is a classic two-player strategy game that involves placing Xs and Os on a 3x3 grid. This project aims to implement Tic Tac Toe using the Pygame library in Python, providing an interactive and visually appealing gaming experience.
The game starts by displaying an empty grid on the Pygame window. Players take turns clicking on the desired cell to place their respective symbols.

Problem Statement

Design and implement a two-player Tic Tac Toe game using the Pygame library in Python. The goal is to create a visually appealing and interactive gaming experience that adheres to the classic rules of Tic Tac Toe.

Aim and Objective

Aim:

The Aim is to provide an enjoyable gaming experience with intuitive controls, visually appealing graphics, and smooth gameplay. The game should be easy to understand and provide a challenging experience for players of all ages.

Objectives:

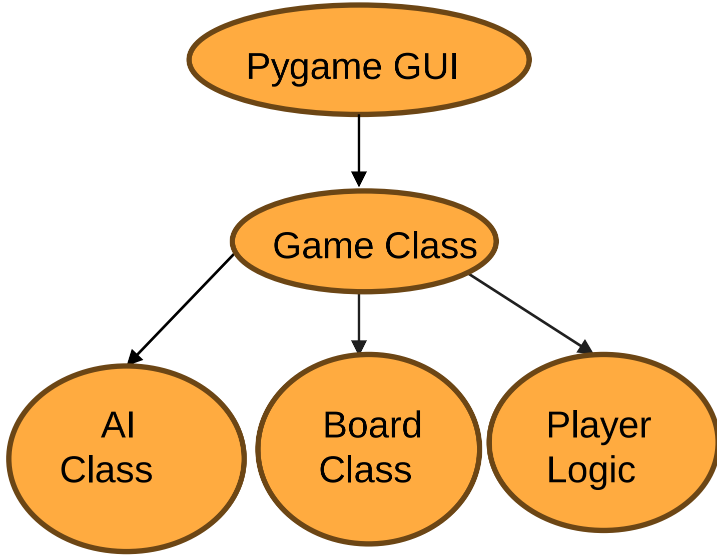
- ➢ Develop an interactive and visually appealing GUI using Pygame.
- ➢ Enable players to make moves using mouse clicks.
- ➢ Implement the underlying logic to determine game outcomes.
- ➢ Organize the code for readability and maintainability.

Proposed Solution

This Python code utilizes Pygame to create a graphical Tic Tac Toe game where players take turns clicking on an empty cell in a 3x3 grid, marking it with 'X' or 'O.' Thinterface updates accordingly until the game is won, tied, or manually closed by the user.e game checks for a winner or a tie after each move, and the graphical

System Architecture

- **Game Board (Board Class):**
    - Manages Tic Tac Toe board state.
    - Tracks player moves.
- **Player Logic (Game Class):**
    - Manages game flow, player turns, and interactions.
    - Handles user input for moves.
    - Controls game state.
- **AI Logic (AI Class):**
    - Implements AI player logic using minimax.
    - Configurable difficulty levels (random or minimax).
- **Constants and Config (constants.py):**
    - Stores program constants (colors, dimensions).
- **Main Loop (main() Function):**
    - Program's main loop.
    - Listens for events, updates game state, and refreshes display.
    - Handles key presses (change mode, restart, AI levels).
- **System Flow:**
    - Initializes, handles events, manages game and AI logic.
    - Renders current game state.
    - Exits program gracefully.



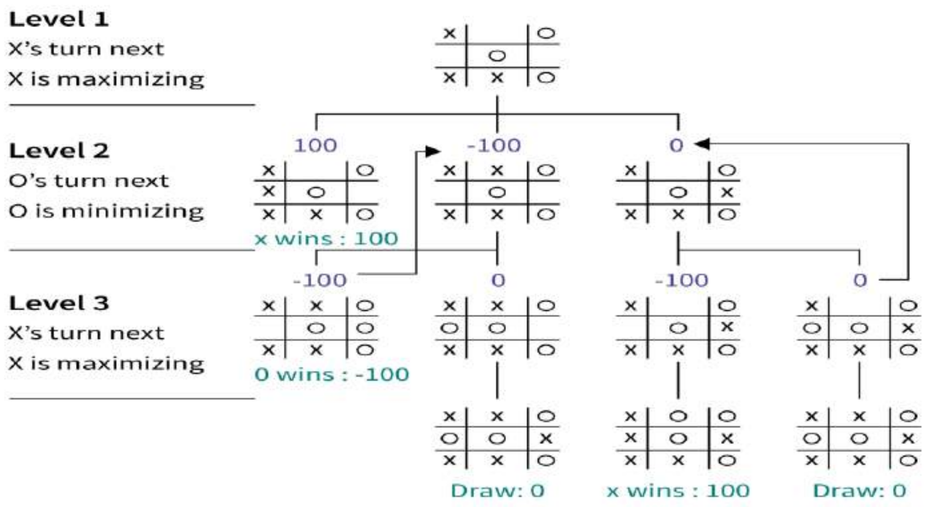
System Deployment Approach

- **Requirements Analysis:**
  - Identify and document the requirements for the Tic Tac Toe game.
  - Define features, functionality, and user interactions.
- **Design:**
  - Define the overall system architecture, including classes and their relationships.
  - Design the user interface using Pygame.
  - Create a plan for managing game state, player input, and AI logic.
- **Implementation:**
  - Write the code for the Tic Tac Toe game based on the design.
  - Implement the Board, Game, AI, and Pygame components.
- **Testing:**
  - Conduct integration testing to ensure seamless interaction between components.
  - Test the game's functionality, including player moves, AI decisions, and win/draw conditions.
- **Debugging:**
  - Identify and fix any bugs or issues found during testing.
  - Debug the code to ensure smooth execution and handle edge cases.

- **Optimization:**
  - Optimize the code for efficiency, especially the AI algorithm.
  - Ensure the game runs smoothly and responds quickly to user inputs.
- **User Interface Refinement:**
  - Refine the Pygame-based user interface for a polished and visually appealing experience.
  - Improve graphics, animations, and overall user interaction.
- **Documentation:**
  - Document the code, including comments and explanations for each component.
  - Create user documentation explaining how to play the game and any configurable options.
- **Deployment:**
  - Package the game for distribution.
  - Provide installation instructions if applicable.
- **Feedback and Iteration:**
  - Collect feedback from users or testing.
  - Iterate on the game based on feedback, addressing any identified issues or suggestions.
- **Maintenance:**
  - Monitor the game for any post-release issues.
  - Perform periodic maintenance, addressing updates or improvements as needed.

Algorithm

The minimax algorithm to make decisions about its moves in the Tic Tac Toe game. The minimax algorithm is a recursive algorithm used for decision-making in two-player games with perfect information, such as Tic Tac Toe.



Conclusion

In conclusion, the development of Tic Tac Toe using Pygame has resulted in a successful implementation of a classic game with an interactive and visually appealing interface. The project focused on creating an enjoyable gaming experience while incorporating key game development concepts and leveraging the Pygame library.

Future Scope

UI/UX Improvements:

•Add a start menu with options to choose between player vs. player and player vs. AI modes.

Score Tracking:

•Keep track of the score for player vs. player and player vs. AI games.

Animations:

•Add animations for moves, like highlighting the winning combination or fading out the board for a smooth transition between moves.

Network Multiplayer:

•Implement network multiplayer functionality, allowing players to play against each other online.

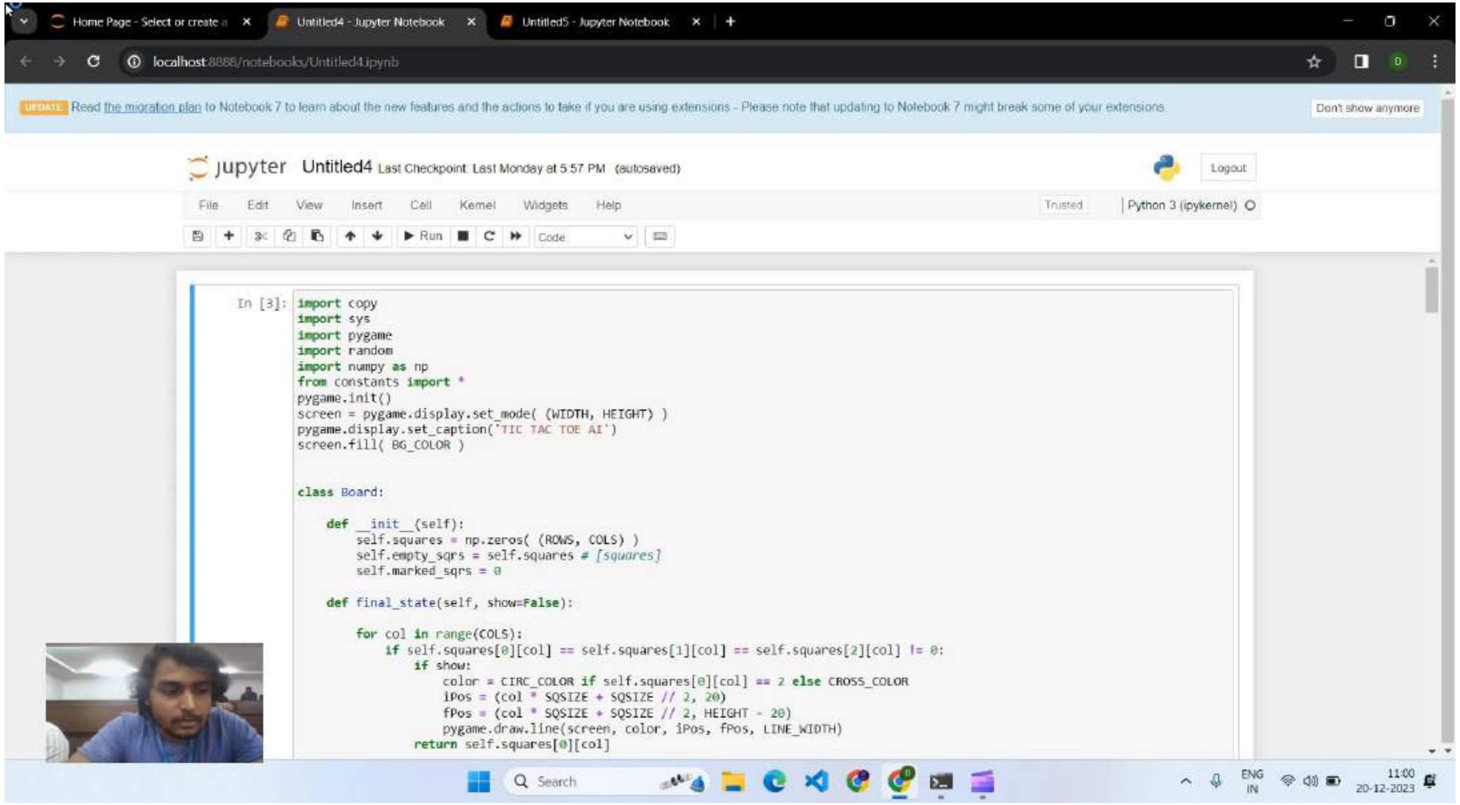
Reference

https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_max_min_problem.htm

https://en.wikipedia.org/wiki/Tic-tac-toe

https://github.com/SamarpanCoder2002/Tic-Tac-Toe-AI

Project Title : Tic Tac Toe Using PyGame


Home Page - Select or create a
Untitled4 - Jupyter Notebook
Untitled5 - Jupyter Notebook
localhost:8888/notebooks/Untitled4.ipynb
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter Untitled4 Last Checkpoint: Last Monday at 5:57 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Search
ENG IN
11:00 20-12-2023

In [3]:

```
import copy
import sys
import pygame
import random
import numpy as np
from constants import *
pygame.init()
screen = pygame.display.set_mode( (WIDTH, HEIGHT) )
pygame.display.set_caption('TIC TAC TOE AI')
screen.fill( BG_COLOR )


class Board:

    def __init__(self):
        self.squares = np.zeros( (ROWS, COLS) )
        self.empty_sqrs = self.squares # [squares]
        self.marked_sqrs = 0

    def final_state(self, show=False):

        for col in range(COLS):
            if self.squares[0][col] == self.squares[1][col] == self.squares[2][col] != 0:
                if show:
                    color = CIRC_COLOR if self.squares[0][col] == 2 else CROSS_COLOR
                    iPos = (col * SQSIZE + SQSIZE // 2, 20)
                    fPos = (col * SQSIZE + SQSIZE // 2, HEIGHT - 20)
                    pygame.draw.line(screen, color, iPos, fPos, LINE_WIDTH)
                return self.squares[0][col]
```

edunet foundation

Thank you!