МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ЛЬВІВСЬКА ПОЛІТЕХНІКА

АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

Task4 Create doxygen documentation

Виконав :

Студент групи КІ-401

Онисько М.М.

Прийняв:

Федак П.Р

2024

Завдання:

1. Розробити Doxygen документацію
2. Створити CI файл для автоматизованого встановлення та генерації документації
3. Оновити Readme.md
4. Додати тег про нову версію
5. Злити створену гілку до develop

Теоретичні відомості:

Doxygen — кросплатформна система документування початкового коду програм, яка підтримує C++, Ci, Objective-C, Python, Java, IDL, PHP, Perl, C#, Фортран, VHDL і, частково, D.

Приклад програми мовою C++

```
class Time {

    public:

        /**
         * Constructor that sets the time to a given value.
         *
         * @param timemillis Number of milliseconds
         *        passed since Jan 1, 1970.
         */
        Time (int timemillis) {
            // the code
        }

        /**
         * Get the current time.
         *
         * @return A time object set to the current time.
         */
        static Time now () {
            // the code
        }
};
```

Doxygen має вбудовану підтримку генерації документації в форматі HTML, LaTeX, man, RTF і XML. Також результати його роботи можуть бути легко конвертовані в CHM, PostScript, PDF.

Для HTML-представлення документації, що розміщується на web-серверах, існує зручний спосіб організації пошуку (за допомогою створюваного Doxygen'ом PHP-модуля) і посилань на зовнішню документацію.

Doxygen використовується в багатьох проектах, в тому числі KDE, Pidgin, Torque Game Engine, AbiWord, Mozilla, FOX toolkit, Crystal Space, Drupal. Є вбудована підтримка в KDevelop.

Doxygen — консольна програма в стилі класичної Unix. Вона працює подібно компілятору, аналізуючи вихідні тексти і створюючи документацію. Додаткові параметри для створення документації можуть читатись із конфігураційного файлу, що має простий текстовий формат.

Для спрощення маніпуляцій з конфігураційним файлом (який містить досить багато налаштувань), існує кілька програм з графічним інтерфейсом: програма doxywizard (реалізована з використанням Qt-3) поставляється разом з Doxygen; програма Doxygate заснована на Qt версії 4. Пізніше doxywizard був переписаний на Qt-4 і проект Doxygate був закритий.

Doxygen генерує документацію на основі набору вихідних текстів і також може бути налаштований для вилучення структури програми з недокументованих вихідних текстів. Можливе складання графів залежностей програмних об'єктів, діаграм класів та вихідних кодів з гіперпосиланнями.

Виконання

Код скрипту для HW частини

```
#include <Arduino.h>
#include <ArduinoJson.h>

const int BOARD_SIZE = 3;
char board[BOARD_SIZE][BOARD_SIZE];
char currentPlayer = 'X';
bool gameOver = false;
int gameMode = 0;

/**
 * @brief Initializes the TicTacToe board and resets game variables.
 * Sets all board positions to empty (' ') and sets the current player to 'X'.
 */
void initializeBoard() {
    for (int i = 0; i < BOARD_SIZE; i++) {
        for (int j = 0; j < BOARD_SIZE; j++) {
            board[i][j] = ' ';
        }
    }
    currentPlayer = 'X';
    gameOver = false;
}

/**
 * @brief Sends a JSON message over Serial.
 *
 * @param type The type of message (e.g., "info", "error", "win_status").
 * @param message The message content.
 */
void sendJsonMessage(const char* type, const char* message) {
    StaticJsonDocument<200> doc;
    doc["type"] = type;
    doc["message"] = message;
    serializeJson(doc, Serial);
    Serial.println();
}

/**
 * @brief Sends the current state of the board over Serial as a JSON message.
```

```
 */
void sendBoardState() {
   StaticJsonDocument<300> doc;
   doc["type"] = "board";
   JsonArray boardArray = doc.createNestedArray("board");
   for (int i = 0; i < BOARD_SIZE; i++) {
      JsonArray row = boardArray.createNestedArray();
      for (int j = 0; j < BOARD_SIZE; j++) {
         row.add(String(board[i][j]));
      }
   }
   serializeJson(doc, Serial);
   Serial.println();
}

/**
 * @brief Checks if the current player has won the game.
 *
 * @return true if the current player has a winning combination, false otherwise.
 */
bool checkWin() {
   for (int i = 0; i < BOARD_SIZE; i++) {
      if (board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) return true;
      if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer) return true;
   }
   if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) return true;
   if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer) return true;
   return false;
}

/**
 * @brief Checks if the game has ended in a draw.
 *
 * @return true if the board is full and there is no winner, false otherwise.
 */
bool checkDraw() {
   for (int i = 0; i < BOARD_SIZE; i++) {
      for (int j = 0; j < BOARD_SIZE; j++) {
         if (board[i][j] == ' ') return false;
      }
   }
   return true;
}

/**
 * @brief Performs a random move for the AI.
 * Places the current player's symbol at a random empty position on the board.
 */
void aiMoveRandom() {
   while (true) {
      int row = random(0, BOARD_SIZE);
      int col = random(0, BOARD_SIZE);
      if (board[row][col] == ' ') {
         board[row][col] = currentPlayer;
         break;  // Exit the loop after a valid move
      }
   }
}

/**
 * @brief Handles an AI vs AI game mode, making random moves until the game is over.
 * Alternates moves between two AI players until a win or draw condition is met.
 */
void handleAiVsAi() {
   while (!gameOver) {
```

```
        if (checkDraw()) {
            sendJsonMessage("win_status", "It's a draw!");
            gameOver = true;
            return;
        }

        aiMoveRandom();  // AI makes a random move

        if (checkWin()) {
            String message = "Player " + String(currentPlayer) + " wins!";
            sendBoardState();
            sendJsonMessage("win_status", message.c_str());
            gameOver = true;
            return;
        }
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';  // Switch players
        sendBoardState();  // Send the board state after each move
    }
}

/**
 * @brief Makes a move for the current player at the specified board position.
 *
 * @param row The row index (0-2).
 * @param col The column index (0-2).
 * @return true if the move is valid and successful, false otherwise.
 */
bool makeMove(int row, int col) {
    if (row >= 0 && row < BOARD_SIZE && col >= 0 && col < BOARD_SIZE && board[row][col] == ' ' &&
!gameOver) {
        board[row][col] = currentPlayer;
        if (checkWin()) {
            String message = "Player " + String(currentPlayer) + " wins!";
            sendJsonMessage("win_status", message.c_str());
            gameOver = true;
        } else if (checkDraw()) {
            sendJsonMessage("win_status", "It's a draw!");
            gameOver = true;
        } else {
            currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
        }
        return true;
    }
    return false;
}

/**
 * @brief Initializes the game and sends a startup message.
 * Sets up Serial communication and initializes the board.
 */
void setup() {
    Serial.begin(9600);
    initializeBoard();
    sendJsonMessage("info", "TicTacToe Game Started");
}

/**
 * @brief Main game loop, reads Serial input and processes commands.
 * Processes moves, resets, and mode changes based on JSON commands from Serial input.
 */
void loop() {
    if (Serial.available() > 0) {
        StaticJsonDocument<200> doc;
        String input = Serial.readStringUntil('\n');
        DeserializationError error = deserializeJson(doc, input);
```

```
    if (!error) {
      const char* command = doc["command"];
      if (strcmp(command, "MOVE") == 0) {
        int row = doc["row"];
        int col = doc["col"];
        if (makeMove(row, col)) {
          sendBoardState();
        } else {
          sendJsonMessage("error", "Invalid move.");
        }
      } else if (strcmp(command, "RESET") == 0) {
        initializeBoard();
        sendJsonMessage("game_status", "Game reset.");
        sendBoardState();
      } else if (strcmp(command, "MODE") == 0) {
        gameMode = doc["mode"];
        String message = "Game mode set to " + String(gameMode);
        sendJsonMessage("game_mode", message.c_str());
        initializeBoard();
        sendJsonMessage("game_status", "Game reset.");
        sendBoardState();
      }

      // AI move logic if applicable
      if (gameMode == 1 && !gameOver && currentPlayer == 'O') {
        aiMoveRandom();  // Make a random move for the AI
        if (checkWin()) {
          String message = "Player " + String(currentPlayer) + " wins!";
          sendJsonMessage("win_status", message.c_str());
          gameOver = true;
        } else if (checkDraw()) {
          sendJsonMessage("win_status", "It's a draw!");
          gameOver = true;
        }
        currentPlayer = 'X';  // Switch back to Player X
        sendBoardState();
      } else if (gameMode == 2 && !gameOver) {
        handleAiVsAi();  // Handle AI vs AI
      }
    }
  }
}
```

Код клієнтської частини

```
import threading
import serial
import serial.tools.list_ports
import json
import tkinter as tk
from tkinter import ttk, scrolledtext
from tkinter import messagebox

class UARTCommunication:
    """
    Class to handle UART communication, including opening ports, sending and receiving messages.
    """

    def __init__(self):
        """
        Initializes the UARTCommunication instance.
        """
        self.ser = None
```

```
    def list_ports(self):
        """
        Lists all available serial ports.

        @return A list of available serial port names.
        """
        return [port.device for port in serial.tools.list_ports.comports()]

    def open_port(self, port, baud_rate=9600):
        """
        Opens a specified serial port with a given baud rate.

        @param port The serial port to open.
        @param baud_rate The baud rate for the port (default is 9600).
        @return A message indicating whether the port was opened successfully or an error occurred.
        """
        try:
            self.ser = serial.Serial(port, baud_rate, timeout=1)
            return f"Connected to {port}"
        except Exception as e:
            self.ser = None
            return f"Error: {e}"

    def send_message(self, message):
        """
        Sends a message over the open serial port in JSON format.

        @param message The message (dictionary) to send.
        @return A message indicating success or an error if sending failed.
        """
        if self.ser and self.ser.is_open:
            try:
                json_message = json.dumps(message)
                self.ser.write((json_message + "\n").encode())
                return f"Sent: {json_message}"
            except Exception as e:
                return f"Error: {e}"
        return "Port not opened"

    def receive_message(self):
        """
        Receives a message from the serial port, attempting to parse it as JSON.

        @return The parsed JSON message if successful, or an error message if receiving failed.
        """
        if self.ser and self.ser.is_open:
            try:
                if self.ser.in_waiting > 0:
                    response = self.ser.readline().decode().strip()
                    if response:
                        json_response = json.loads(response)
                        return json_response
            except json.JSONDecodeError:
                return "Error: Invalid JSON received"
            except Exception as e:
                return f"Error: {e}"
        return "Port not opened"


def update_game_board(board, buttons):
    """
    Updates the GUI game board with the current board state.

    @param board A 2D list representing the game board.
```

```
    @param buttons The GUI button widgets for each cell in the game board.
    """
    for i in range(3):
        for j in range(3):
            buttons[i][j].config(text=board[i][j])


def send_move(uart, row, col):
    """
    Sends a MOVE command with the selected row and column to the UART.

    @param uart The UARTCommunication instance for sending the command.
    @param row The row of the move.
    @param col The column of the move.
    """
    message = {"command": "MOVE", "row": row, "col": col}
    uart.send_message(message)


def set_mode(uart, mode):
    """
    Sends a MODE command to the UART to set the game mode.

    @param uart The UARTCommunication instance for sending the command.
    @param mode The game mode to set (e.g., 0 for User vs User).
    """
    message = {"command": "MODE", "mode": mode}
    uart.send_message(message)


def reset_game(uart):
    """
    Sends a RESET command to the UART to reset the game.

    @param uart The UARTCommunication instance for sending the command.
    """
    message = {"command": "RESET"}
    uart.send_message(message)


def auto_receive(uart, buttons, output_text, root):
    """
    Periodically checks for incoming messages on the UART and updates the GUI accordingly.

    @param uart The UARTCommunication instance for receiving messages.
    @param buttons The GUI button widgets for each cell in the game board.
    @param output_text The text area for displaying received messages.
    @param root The main tkinter root window for scheduling periodic checks.
    """
    try:
        if uart.ser and uart.ser.is_open:
            response = uart.receive_message()
            if response and response != "Port not opened":
                if isinstance(response, dict):
                    if "board" in response:
                        update_game_board(response["board"], buttons)
                    else:
                        output_text.insert(tk.END, f"Game status: {response['message']}\n")

                    if response.get("type") == "win_status":
                        thread = threading.Thread(target=messagebox.showinfo, args=("Win Status",
                                                                                    response.get("message")))
                        thread.start()

                else:
```

```
                output_text.insert(tk.END, f"Received: {response}\n")
            output_text.see(tk.END)
    except Exception as e:
        output_text.insert(tk.END, f"Error: {str(e)}\n")
    root.after(100, lambda: auto_receive(uart, buttons, output_text, root))


def start_gui():
    """
    Initializes and runs the GUI for the Tic-Tac-Toe game, handling UART communication and game interactions.
    """
    uart = UARTCommunication()

    root = tk.Tk()
    root.title("TicTacToe Game Interface")

    # GUI components for port selection and status
    port_label = tk.Label(root, text="Select Port:")
    port_label.grid(row=0, column=0, padx=10, pady=10)
    port_var = tk.StringVar()
    port_combobox = ttk.Combobox(root, textvariable=port_var, values=uart.list_ports(), state="readonly")
    port_combobox.grid(row=0, column=1, padx=10, pady=10)

    def open_port_callback():
        """
        Opens the selected port and starts auto-receive if successful.
        """
        status = uart.open_port(port_var.get())
        status_label.config(text=status)
        if "Connected" in status:
            auto_receive(uart, buttons, output_text, root)
        else:
            output_text.insert(tk.END, f"Failed to connect: {status}\n")

    open_button = tk.Button(root, text="Open Port", command=open_port_callback)
    open_button.grid(row=0, column=2, padx=10, pady=10)

    # GUI game board buttons
    buttons = [[None for _ in range(3)] for _ in range(3)]
    for i in range(3):
        for j in range(3):
            button = tk.Button(root, text=" ", width=10, height=3,
                               command=lambda row=i, col=j: send_move(uart, row, col))
            button.grid(row=i + 1, column=j, padx=5, pady=5)
            buttons[i][j] = button

    # Game mode selection components
    mode_label = tk.Label(root, text="Select Game Mode:")
    mode_label.grid(row=4, column=0, padx=10, pady=10)
    mode_var = tk.StringVar(value="User vs User")
    mode_combobox = ttk.Combobox(root, textvariable=mode_var,
                                 values=["User vs User", "User vs AI", "AI vs AI"],
                                 state="readonly")
    mode_combobox.grid(row=4, column=1, padx=10, pady=10)

    def set_mode_callback():
        """
        Sets the game mode based on the user's selection.
        """
        mode_index = mode_combobox.current()
        set_mode(uart, mode_index)
        status_label.config(text=f"Game mode set to {mode_combobox.get()}")

    mode_button = tk.Button(root, text="Set Mode", command=set_mode_callback)
    mode_button.grid(row=4, column=2, padx=10, pady=10)
```

```
    # Reset button for resetting the game
    reset_button = tk.Button(root, text="Reset", command=lambda: reset_game(uart))
    reset_button.grid(row=5, column=1, padx=10, pady=10)

    # Output text area for displaying messages
    output_text = scrolledtext.ScrolledText(root, width=50, height=10, wrap=tk.WORD)
    output_text.grid(row=6, column=0, columnspan=3, padx=10, pady=10)

    # Status label for connection information
    status_label = tk.Label(root, text="Status: Not connected", fg="blue")
    status_label.grid(row=7, column=0, columnspan=3, padx=10, pady=10)

    root.mainloop()


if __name__ == "__main__":
    start_gui()
```

Автоматизація

DoxygenCreateWindows.ps – Лістинг CI скрипту

```
# PowerShell Script to Install Doxygen and Generate Documentation

# Define paths
$doxygenInstallerUrl = "https://doxygen.nl/files/doxygen-1.12.0-setup.exe" # Replace with the latest version if needed
$doxygenInstallerPath = "$env:TEMP\doxygen-setup.exe"
$projectDir = Join-Path -Path $PSScriptRoot -ChildPath ".." # Replace with the path to your project
$outputDir = "$projectDir\docs" # Path for generated documentation

# Step 1: Check if Doxygen is installed
Write-Output "Checking if Doxygen is installed..."
$doxygenPath = (Get-Command "doxygen" -ErrorAction SilentlyContinue).Source

if (-not $doxygenPath) {
    Write-Output "Doxygen not found. Downloading and installing Doxygen..."

    # Download Doxygen installer
    Invoke-WebRequest -Uri $doxygenInstallerUrl -OutFile $doxygenInstallerPath -UseBasicParsing

    # Run the installer silently
    Start-Process -FilePath $doxygenInstallerPath -ArgumentList "/S" -Wait

    # Confirm installation
    $doxygenPath = (Get-Command "doxygen" -ErrorAction SilentlyContinue).Source
    if (-not $doxygenPath) {
        Write-Output "Doxygen installation failed. Please install it manually."
        exit 1
    }
     # Add Doxygen to PATH
    $doxygenPath = "C:\Program Files\doxygen\bin" # Default installation path, adjust if different
    [System.Environment]::SetEnvironmentVariable("Path", $env:Path + ";$doxygenPath",
[System.EnvironmentVariableTarget]::Machine)

    Write-Output "Doxygen installed successfully."
} else {
    Write-Output "Doxygen is already installed at $doxygenPath."
}

# Step 2: Create Doxygen configuration file if not exists
$doxyfilePath = "$projectDir\Doxyfile"
if (-not (Test-Path $doxyfilePath)) {
    Write-Output "Generating Doxygen configuration file..."
    Start-Process -FilePath "doxygen" -ArgumentList "-g $doxyfilePath" -Wait
}

# Step 3: Update configuration file for your project settings
```

```
(Get-Content $doxyfilePath) -replace 'OUTPUT_DIRECTORY.*', "OUTPUT_DIRECTORY = $outputDir" | Set-Content $doxyfilePath
(Get-Content $doxyfilePath) -replace 'INPUT.*', "INPUT = $projectDir" | Set-Content $doxyfilePath
(Get-Content $doxyfilePath) -replace 'RECURSIVE.*', "RECURSIVE = YES" | Set-Content $doxyfilePath

# Step 4: Run Doxygen to generate documentation
Write-Output "Generating documentation..."
Start-Process -FilePath "doxygen" -ArgumentList "$doxyfilePath" -Wait

Write-Output "Documentation generation complete. Output available at $outputDir."
```

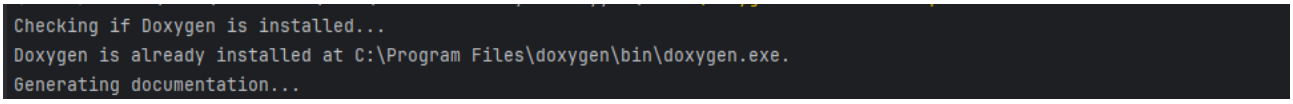

Рис.1 – Виконання CI скрипту

Рис.2 – Вигляд документації

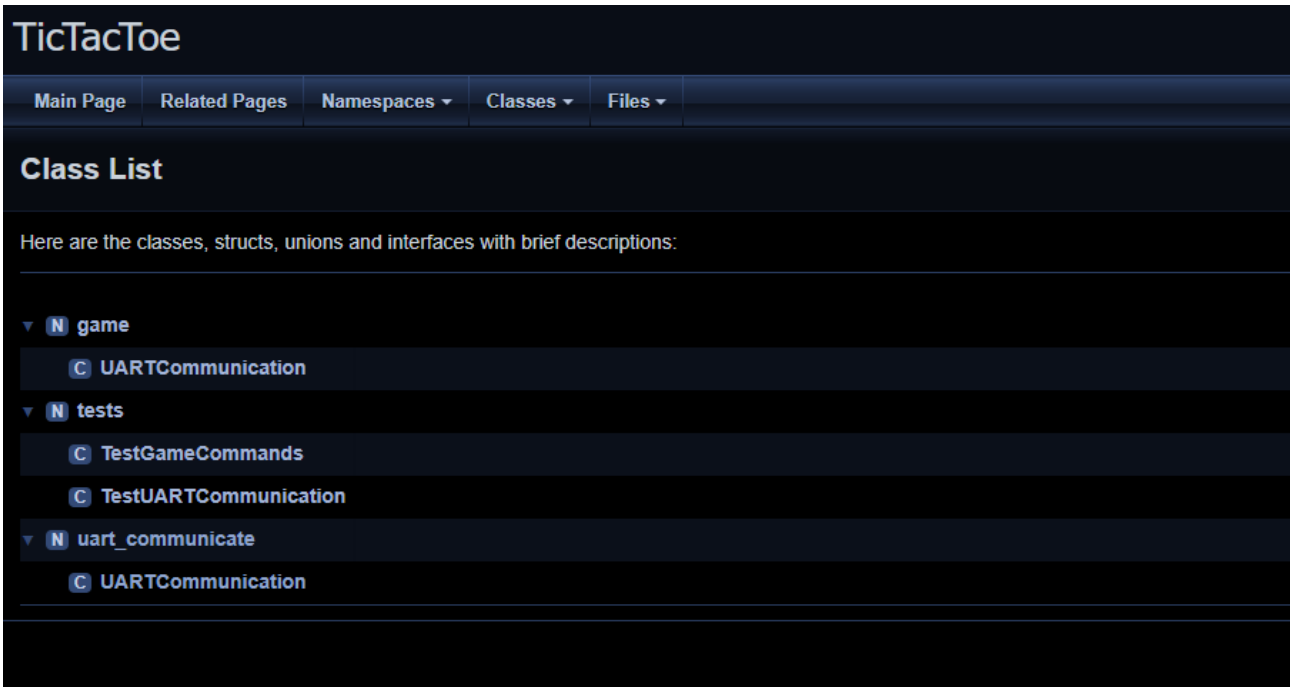

Рис.3 – Вигляд списку класів в документації

Висновки

На лабораторній роботі я додав документацію до проекту, додавши коментарі та згенерував Doxygen файл. Зробив скрипт для автоматизації встановлення Doxygen та генерації документації.

Посилання

1. https://www.arduino.cc/

2. https://uk.wikipedia.org/wiki/Doxygen

3. https://docs.python.org/uk/3/library/tkinter.html

4. https://pyserial.readthedocs.io/en/latest/pyserial.html

5. https://stackoverflow.com/questions/58622/how-to-document-python-code-using-doxygen