3D Tetris on a 3x3x12 LED Matrix

Welcome to the 3D Tetris project! This repository documents the creation of a 3x3x12 LED matrix that enables users to play a fully functional 3D Tetris game. The project leverages an ESP32-based development board (ESPduino) with integrated web server functionality for remote control.

Project Overview

This project was developed as part of the **"Applied Microcontroller Technology"** module and integrates hardware assembly, software programming, and web-based GUI design. Inspired by the game available at 3DTetris.de, the system enables interactive gameplay through a wireless web interface.

Key features include:

- A custom-built 3x3x12 RGB LED matrix.
- ESP32 microcontroller for controlling LEDs and hosting a local web server.
- A responsive web interface for gameplay and real-time interactions.
- Open-source software written in C, HTML, CSS, and JavaScript.

Features

Hardware

- **ESPduino Controller**: Combines the ESP32 Wi-Fi chip with Arduino-compatible GPIO layout.
- **3x3x12 LED Matrix**: Comprised of 108 individually addressable PL9823 RGB LEDs.
- **Custom Housing**: 3D-printed for modularity and ease of assembly.
- **Power Supply**: Powered via USB-C with an integrated on/off switch.

Software

- **LED Control**: Powered by the FastLED library for efficient matrix management.
- **Game Logic**: Implements Tetris gameplay mechanics including rotation, collision detection, and line clearing.
- **Web Interface**: HTML/CSS-based GUI hosted via an onboard web server, with multi-language support.

Gameplay

- Control Tetrominoes (Tetris blocks) through the GUI.
- Real-time rendering of blocks on the LED matrix.
- Score tracking and high-score leaderboard.
- Adjustable levels and difficulty scaling.
- High-score table to save and display the top 10 scores.
- Difficulty selection for custom gameplay experiences.

Getting Started

Prerequisites

Assembly

- Circuit: The LED matrix is powered and controlled through a single GPIO pin of the ESPduino. A 5V USB-C power supply is used for both the matrix and the ESPduino.
- LED Matrix: LEDs are arranged in a 3x3x12 grid and tested individually for functionality. Custom fixtures were used to ensure precision during soldering.
- 3D printed Case: The case, 3D-printed using PLA, features dedicated mounts for the ESPduino, USB ports, and optional counterweights to enhance stability.
- How the Code Works: The ESP32 Tetris game is explained using a flowchart, detailing initialization, web interface, user input processing, game logic, scoring, and game over handling.

Software

- **Arduino IDE**: Download and install the Arduino IDE.
- **ESP32 Board Manager**: Add the ESP32 board manager by following this guide.
- **Libraries**: Install the following libraries:
  - Via Arduino IDE Library Manager:
    - `FastLED`
    - `WiFi`
    - `Preferences`
  - **ESPForm Library**:
    - Download the library from the ESPForm GitHub repository.
    - Extract the downloaded ZIP file.
    - Copy the extracted folder into the `libraries` directory of your Arduino IDE sketchbook folder.

Installation

1. **Clone the Repository**
   Open a terminal and run the following command to clone the repository:

   ```
   git clone https://github.com/18Markus1984/3D-Tetris.git
   ```

2. **Open the Project in Arduino IDE**
   Navigate to the cloned folder and open the `.ino` file in the Arduino IDE.

3. **Configure the ESP32 Board**
   - Go to **Tools** > **Board** and select **ESP32 Dev Module**.
   - Enable **PSRAM** under **Tools** > **PSRAM**.

4. **Upload the Sketch**
   - Ensure the correct COM port is selected under **Tools** > **Port**.
   - Click the **Upload** button to flash the code onto the ESPduino.

5. **Connect to the ESP32's Wi-Fi Network**
   - On your computer or mobile device, connect to the Wi-Fi network:
     - **SSID**: `3D-Tetris`
     - **Password**: `12345678`

6. **Access the Web Interface**
   - Open a web browser and navigate to:
     - http://192.168.4.1

You can now enjoy 3D Tetris using the interactive web interface!

How It Works

Code Flow Chart

```
graph TD;
    Start["ESP32 starts"] --> InitWiFi["Initialize WiFi"];
    InitWiFi --> LoadWebPage["Load webpage from html.h"];
    LoadWebPage --> WaitForClient["Wait for client requests"];

    WaitForClient -->|Start button pressed| StartGame["Start game"];
    StartGame --> GameLoop["Game loop"];

    GameLoop -->|Movement input received| UpdatePosition["Update piece position"];
    UpdatePosition --> GameLoop;

    GameLoop -->|Rotation input received| RotatePiece["Rotate piece"];
    RotatePiece --> GameLoop;

    GameLoop -->|Piece falls| CheckCollision["Check collision"];
    CheckCollision -->|Collision detected| PlacePiece["Place piece"];
    PlacePiece --> CheckLines["Check for complete lines"];
    CheckLines -->|Lines found| RemoveLines["Remove lines"];
    RemoveLines --> UpdateScore["Update score"];
    UpdateScore --> GameLoop;

    CheckLines -->|No lines| GameLoop;

    GameLoop -->|Game over| GameOver["Game Over"];
    GameOver --> WaitForClient;
```

LED Matrix

- **Construction**: LEDs are arranged in a 3x3x12 format, connected using PL9823-compatible drivers.
- **Control**: Each LED is addressed individually using a mapped index for 3D coordinates.

Web Interface

- **Languages Supported**: English, German, French, Spanish, Italian.
- **Functions**:
  - Rotate blocks around X, Y, Z axes.
  - Shift blocks along the XY-plane.
  - Drop blocks into place.
- **Additional Features**:
  - Live preview of the next Tetromino.
  - Display of current score and level.
  - **High-Score Table**: Saves and displays the top 10 scores along with player names. Extra entries are shown but not stored persistently.
  - **Difficulty Selection**: Choose between multiple difficulty levels to adjust the gameplay experience.

Results

The project met its primary objectives, showcasing:

1. A functional 3D Tetris game on a custom LED matrix.
2. An intuitive and accessible web-based control interface.
3. High scalability for future enhancements, including battery power and improved stability.

Future Improvements

- Reduce input latency for smoother gameplay.
- Enhance matrix stability to minimize movement during transport.
- Add battery support for portability.
- Optimize collision detection and rotation algorithms to avoid edge-case errors.

Acknowledgments

This project was developed by Marvin Heins, Markus Schmidt, and Leonard Holz-Diehl. Special thanks to the open-source community and the developers of the FastLED library.

License