Tetris書面報告

一、系統摘要 / 功能 / 架構

本系統為一款使用 HTML5 Canvas 製作的俄羅斯方塊遊戲，配合 Node.js + Express 架設本地伺服器，實現跨裝置可同步的排行榜功能。

系統用途：

1. 提供使用者單人遊玩經典俄羅斯方塊遊戲
2. 支援實時分數計算與等級提升
3. 結束遊戲後可輸入暱稱並上傳排行榜資料
4. 顯示前 10 高分排行榜

系統架構：

前端介面：HTML + CSS + JavaScript
音樂與 UI 控制：Canvas 顯示、事件綁定
後端架構：Node.js + Express 伺服器
資料儲存：JSON 檔案模擬本地資料庫
通訊方式：Fetch API (GET/POST)

二、開發平台與工具

前端：

HTML5 + CSS3
JavaScript ES6
使用 Canvas API 繪製遊戲圖形
音效與背景音樂使用 audio 控制器與自製 UI 音樂面板

函式庫：

使用 fetch() 串接後端 REST API

開發環境：

Windows 10 作業系統
使用 Visual Studio Code 編輯器
使用 Live Server 插件進行前端測試

後端伺服器資訊：

Node.js v20+
使用 Express 框架啟動 REST API
本地伺服器開啟於 http://140.113.65.41:3000/
排行榜資料儲存在本地 rank.json

三、程式邏輯與檔案說明

檔案結構：

`server.js`（伺服器程式）
`rank.json`（排行榜資料）
`public/`
index.html（主畫面）
main.js（遊戲邏輯）
style.css（畫面設計）
sounds/ (音效資料夾)
Musics/（背景音樂資料夾）
pictures/（控制圖片與背景）

核心功能程式說明：

main.js:

> 遊戲邏輯

先畫出場地的格線
再從start()開始
drawGrid()畫出主圖上的格線
drawGridsmall()劃出下一個的格線
drawShapesmall()劃出下一個的方塊
然後進入遊戲主要程式`update()`

```
function start() {
  context.fillStyle = "#000";
  context.fillRect(0, 0, COLS, ROWS);
  drawGrid();
  drawGridsmall();
  drawShapesmall(nextShape, 0, 0, "#1F7F6C");
  gameLoop = setInterval(update, 500);
}
```

update()

主要是用collision()來辨別下一個方向移動會不會撞到

沒有撞到就posY++來下墜

撞到就merge()到虛擬的board上

用clearLines()來判定是否要消除

12行再判斷當方塊出現時是否就collision了，若是，則遊戲結束。

20~32行則是每次下降要重劃一次方塊及場地

Levelup()則是在全域變數score加100會升一級和速度變快

```
function update() {
    if(!paused){
        if (!collision(currentShape, posX, posY + 1)) posY++;
        else {
            merge(currentShape, posX, posY);
            clearLines();
            currentShape = nextShape;
            nextShape = randomShape();
            drawShapesmall(nextShape, 0, 0, "#1F7F6C");
            posX = 3;
            posY = 0;
            if (collision(currentShape, posX, posY)) {
              const elapsedTime = Math.floor((Date.now() - startTime) / 1000)
              playSound("gameover");
              document.getElementById("gameOverPanel").classList.remove("hid
              clearInterval(gameLoop);

            }
        }
        context.fillStyle = "#000";
        context.fillRect(0, 0, COLS, ROWS);
        drawGrid();
        for (let y = 0; y < ROWS; y++) {
            for (let x = 0; x < COLS; x++) {
            if (board[y][x]) drawBlock(x, y, "#1F7F6C");
            }
        }
        drawShape(currentShape, posX, posY, "#1F7F6C");
        document.getElementById("scoreDisplay").textContent = `Score: ${score
        const elapsedTime = Math.floor((Date.now() - startTime) / 1000);
        document.getElementById("timerDisplay").textContent = `Time: ${elapse
        LevelUp();
    }
}
```

> 排行榜

(gameover)

當遊戲結束出現面板，輸入名字(不能與資料庫重複)後

會用saveScoreToRank()來fetch到json資料庫

而在submitScore()後會停兩秒看排行榜showRank()

```
function saveScoreToRank(score, time, name) {
  const isValid = /^[\u4e00-\u9fa5a-zA-Z0-9_]{1,15}$/.test(name);
  if (!isValid) {
    alert("⚠️ 名字不合法！請使用中英文、數字或底線，最多15字");
    return;
  }


  fetch('http://140.113.65.41:3000/rank')
    .then(res => res.json())
    .then(data => {
      const nameExists = data.some(entry => entry.name === name);
      if (nameExists) {
        alert("此名字已經存在排行榜中，請換一個！");
        return;
      }

      return fetch('http://140.113.65.41:3000/rank', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({ name, score, time })
      });
    })
    .then(res => res?.json?.())
    .then(data => {
      if (data) {
        console.log("儲存成功", data);

        gameElements.forEach(el => el.style.display = "none");
        document.getElementById("gameOverPanel").classList.add("hidden");
        showRank();
        rankPanel.classList.remove("hidden");

        setTimeout(() => {
          rankPanel.classList.add("hidden");
          gameElements.forEach(el => el.style.display = "");
          resetGame();
        }, 3000);
      }
    })
    .catch(err => {
      console.error("發生錯誤：", err);
    });
}
```

(點擊左側排行榜)

會暫停遊戲paused = false

showRank()再次點擊排行榜會消失並按下右邊開始按鈕開始

```
function showRank() {
  fetch('http://140.113.65.41:3000/rank')
    .then(res => res.json())
    .then(rankData => {
      const rankList = document.getElementById("rankList");
      rankList.innerHTML = "";
      rankData.forEach((entry, index) => {
        const li = document.createElement("li");
        li.textContent = `${index + 1}. ${entry.name} - ${entry.score}分 (${e
        rankList.appendChild(li);
      });
    });
}
```

音樂與音效

音樂部分:
播放背景音樂 player(再點下任意鍵就會開始撥放)
支援音樂清單切換、音量調整與目前播放顯示
togglePlay() 可手動播放 / 暫停

```
//music panel-----------------------------------------------------
const musicPanel = document.getElementById("musicPanel");
const musicList = document.getElementById("musicList");
const currentMusicText = document.getElementById("currentMusic");
const volumeControl = document.getElementById("volumeControl");


let currentTrack = 0;
player.src = musicFiles[currentTrack];
player.volume = 0.5;

// 音樂 UI 初始化
function initMusicList() {
  musicFiles.forEach((file, index) => {
    const li = document.createElement("li");
    const name = file.split("/").pop().replace(".mp3", "");
    li.textContent = name;
    li.addEventListener("click", () => {
      currentTrack = index;
      player.src = musicFiles[currentTrack];
      player.play();
      updateCurrentMusicText();
    });
    musicList.appendChild(li);
  });
  updateCurrentMusicText();
}

function updateCurrentMusicText() {
  const name = musicFiles[currentTrack].split("/").pop().replace(".mp3", "");
  currentMusicText.textContent = `正在播放：${name}`;
}

function toggleMusic() {
  musicPanel.classList.toggle("hidden");
}

function togglePlay() {
  if (player.paused) {
    player.play();
  } else {
    player.pause();
  }
}
```

可以在start button按下後直接播放

```
document.getElementById("startBtn").addEventListener("click", () => {
  // 隱藏按鈕
  document.getElementById("startBtn").style.display = "none";

  // 播放背景音樂
  player.play().then(() => {
    console.log("✅ 背景音樂播放成功");
  }).catch(err => {
    console.warn("⚠️ 音樂播放失敗", err);
  });
  start();
  // 開始主遊戲迴圈
  startTime = Date.now(); // 重設開始時間
  score = 0;
});
```

音效部分:
clear, move, gameover 使用 Audio 物件播放本地音效檔
執行則是在上左右鍵按下及gameover時發生。

```

const sounds = {
    clear: new Audio("sounds/clear.mp3"),
    move: new Audio("sounds/move.mp3"),
    gameover: new Audio("sounds/gameover.mp3"),
};
sounds.clear.volume = 0.05;
sounds.move.volume = 0.05;
sounds.gameover.volume = 0.2;

function playSound(name) {
    if (sounds[name]) {
      sounds[name].currentTime = 0;
      sounds[name].play();
    }
```

> 按鍵

鍵盤方向鍵：上下左右控制方塊移動、旋轉

觸控按鈕：模擬按鍵事件 handleInput()

可切換 pause 狀態暫停遊戲

```
const pauseBtn = document.getElementById("btnPause");
let paused = false;

document.getElementById("btnUp").addEventListener("click", () => handleInput
document.getElementById("btnDown").addEventListener("click", () => handleInpu
document.getElementById("btnLeft").addEventListener("click", () => handleInpu
document.getElementById("btnRight").addEventListener("click", () => handleInp
pauseBtn.addEventListener("click", () => togglePause());

function handleInput(key) {
   document.dispatchEvent(new KeyboardEvent('keydown', { key }));
}

function togglePause() {
    paused = !paused;
    // 自行設計 paused 時不執行 loop 邏輯
}
```

```
document.addEventListener("keydown", (event) => {
    if(!paused){
        if (event.key === "ArrowLeft" && !collision(currentShape, posX - 1, p
          posX--;
          playSound("move");
        }
        else if (event.key === "ArrowRight" && !collision(currentShape, posX
            posX++;
            playSound("move");
        }
        else if (event.key === "ArrowDown" && !collision(currentShape, posX,
            posY++;
        }
        else if (event.key === "ArrowUp") {
            const rotated = rotate(currentShape);
            if (!collision(rotated, posX, posY)) currentShape = rotated;
            playSound("move");
        }
        updatelow();
        clearLines();
    }
});
```

Server 與資料儲存

server.js

使用 Node.js + Express 建立伺服器，功能如下：

1. 提供靜態檔案路徑
   `app.use(express.static(path.join(__dirname, 'public')));`

2. 在網頁載入時開啟index.html
   `app.get('/')`

3. 提供 `rank get (app.get)` 、 `POST API(app.post)`

4. POST 時自動將新紀錄寫入 rank.json
   `app.use((req, res, next)`

```
const express = require('express');
const fs = require('fs');
const cors = require('cors');//cross origin resource sharing對不同來源的網站的請求
const bodyParser = require('body-parser');
const path = require('path');

const app = express();
const PORT = 3000;//序列阜

app.use(cors());
app.use(bodyParser.json());

app.use(express.static(path.join(__dirname, 'public')));

// 排行榜 API
app.get('/rank', (req, res) => {
  const data = JSON.parse(fs.readFileSync('rank.json'));
  const top10 = data.sort((a, b) => b.score - a.score).slice(0, 10);
  res.json(top10);
});

app.post('/rank', (req, res) => {
  try {
    const newEntry = req.body;
    console.log("收到資料：", newEntry);

    const data = JSON.parse(fs.readFileSync('rank.json'));
    data.push(newEntry);
    fs.writeFileSync('rank.json', JSON.stringify(data, null, 2));//把更新後的資
    res.status(201).json({ message: '儲存成功', entry: newEntry });
  } catch (err) {
    console.error("儲存失敗：", err);
    res.status(500).json({ message: '儲存失敗', error: err.message });
  }
});

app.listen(PORT, () => {
  console.log(`✅ Server running at http://localhost:${PORT}`);
});

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});
app.use((req, res, next) => {
  console.log(`💡 收到請求：${req.method} ${req.url}`);
  next();
});
```

rank.json

使用 JSON 格式儲存

包含排行榜前 10 筆紀錄：名字、分數、時間

由前端用 fetch() 調用取得資料並顯示

四、系統特色與個人心得

系統亮點：

音樂面板支援切換、播放暫停、音量調整與歌曲名稱顯示
排行榜儲存採 REST API，具備基本資料驗證與防重複
提供完整重新開始機制，自動顯示排行榜後重啟遊戲

心得：

這是我第一次從零開始完成一個具備前後端功能的小型系統專案。過程中，雖然是第一次撰寫遊戲邏輯，但實作起來意外地順利，反而是在資料儲存這塊遇到了不少挑戰。一開始我先使用本地網頁的 localStorage 測試排行榜功能，等整體流程跑順之後才改成使用 Node.js 搭配本地伺服器與 JSON 檔案進行儲存，這中間碰到了許多設定上的問題，也讓我深刻體會到「前後端整合」的實作細節。
這次作業讓我對網頁全端開發有了更清晰的理解，也燃起了我對遊戲開發與前端互動設計的興趣。

五、補充資訊

可支援的音樂格式為 .mp3
可使用手機或其他電腦，只要 IP 能連進伺服器，就能查看與提交排行榜