מדריך התקנה: ספריית Raylib

Raylib היא ספרייה פשוטה וקלה לשימוש לפיתוח משחקים ואפליקציות גרפיות בשפת C. במדריך זה נלמד כיצד להתקין את Raylib בסביבת הפיתוח שלנו כדי שנוכל להשתמש בה בפרויקט הגמר שלנו.

שלבי ההתקנה וההגדרה

תהליך ההתקנה וההגדרה מורכב מכמה שלבים:

1. התקנת ספריית Raylib דרך מנהל החבילות של MSYS2.
2. הגדרת קבצי JSON ב-VS Code כך שיכיר את הספרייה ויוכל לקמפל פרויקטים המשתמשים בה.

שלב 1: התקנת ספריית Raylib דרך MSYS2 UCRT64

1. **פתיחת טרמינל MSYS2 UCRT64:**
   - לחצו על כפתור "התחל" במערכת ההפעלה Windows.
   - הקלידו MSYS2 UCRT64.
   - לחצו על האפליקציה שמופיעה כדי לפתוח את חלון הטרמינל (שורת הפקודה).
2. **התקנת Raylib באמצעות Pacman:** בחלון הטרמינל שנפתח, הקלידו את הפקודה הבאה ולחצו Enter

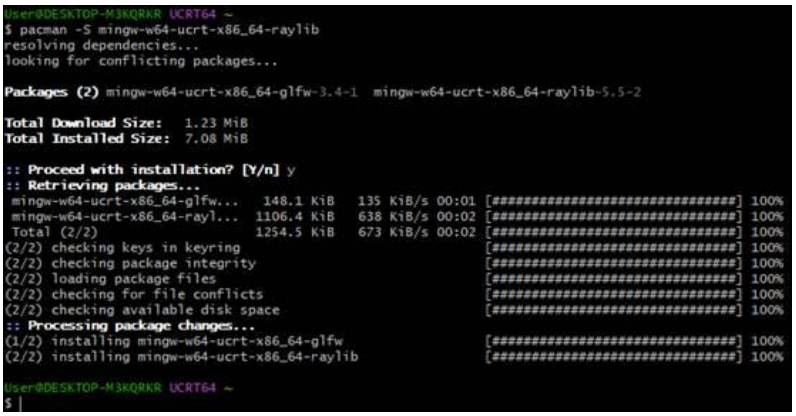
```
Unset
pacman -S mingw-w64-ucrt-x86_64-raylib
```

תזכורת - לא ניתן להשתמש בקיצור מקלדת בעת ההעתקה לכן יש לבצע את ההדבקה של שורת הקוד על-ידי העתק והדבק עם העכבר.

הטרמינל יציג מידע על החבילות שהוא עומד להוריד ולהתקין. הוא ישאל אתכם:

Proceed with installation? [Y/n].

הקלידו y (קיצור של yes) ולחצו Enter כדי לאשר את ההתקנה. ההתקנה תתחיל. המתינו בסבלנות עד שהתהליך יסתיים והסמן יחזור לשורת הפקודה הרגילה.



הסבר:

- pacman: זהו מנהל החבילות של MSYS2 (בדומה ל-apt ב-Linux או Homebrew ב-macOS).
- S-: אומר ל-pacman לסנכרן (להתקין או לעדכן) חבילות.
- mingw-w64-ucrt-x86_64-raylib: זהו השם המלא של חבילת Raylib המותאמת לסביבת UCRT64 של MinGW (הקומפיילר שבו אנו משתמשים).

שלב 2: הגדרת VS Code

כדי ש-VS Code יוכל לקמפל ולהריץ פרויקטים המשתמשים ב-Raylib, עלינו לעדכן שני קבצי הגדרות בפרויקט שלנו: tasks.json (להגדרות בנייה) ו-c_cpp_properties.json (להגדרות IntelliSense וזיהוי קבצים).

חשוב: שלבים אלו מבוצעים *בתוך תיקיית הפרויקט שלכם* ב-VS Code. אם אין לכם עדיין פרויקט, צרו והגדירו אותו על פי מדריך עבודה עם VS Code.

2.1 עדכון קובץ tasks.json (הגדרות בנייה)

קובץ זה מגדיר לקומפיילר (GCC) כיצד לבנות (לקמפל ולקשר) את הפרויקט שלכם. עלינו להוסיף לו את הדגלים (flags) הדרושים לקישור עם ספריית Raylib.

יש לפתוח את הקובץ tasks.json ולהוסיף את הדגלים המצויינים מטה

- אם הקובץ קיים המשיכו למטה, רק **אם לא נוצר** לכם אוטומטית קובץ tasks.json בפרויקט (בתוך תיקיית .vscode) -ניתן להעזר בשלב 6 במדריך עבודה עם VS Code לשם כך, כלומר לפתוח את קובץ c (אפשר ריק) ואז ללחוץ על ctrl+shift+p, לכתוב Tasks: Configure Task ואז בחירה של הקומפיילר המתאים.

הוספת דגלי קישור (Linker Flags) ל-Raylib:

- בתוך קובץ tasks.json, אתרו את החלק של "args": [...]. זהו מערך הארגומנטים המועברים לקומפיילר.

בסוף הפרמטרים של ה-args, לפני הסוגר המרובע הסוגר (כלומר,]), הוסיפו את השורות הבאות. שימו לב לפסיקים בין כל ארגומנט:

```
Unset

"args": [
    ...
    "-lraylib",
    "-lopengl32",
    "-lgdi32",
    "-lwinmm",
    "-static-libgcc"
],
```

הסבר הדגלים:

- lraylib: מקשר את ספריית Raylib עצמה.
- lopengl32: מקשר את ספריית Raylib (OpenGL משתמשת בה לגרפיקה).
- lgdi32: מקשר את ספריית GDI32 של Windows (לפעולות גרפיות מסוימות).
- lwinmm: מקשר את ספריית Windows Multimedia (לסאונד וטיימרים).
- static-libgcc: מקשר את ספריות ה-runtime של GCC באופן סטטי, מה שיכול למנוע בעיות תלות בקבצי DLL במחשבים אחרים.

2.2 עדכון/יצירת קובץ c_cpp_properties.json (הגדרות IntelliSense)

קובץ זה עוזר ל-VS Code להבין את מבנה הפרויקט שלכם, לאתר קבצי header (כמו raylib.h) ולספק השלמה אוטומטית וניתוח קוד (IntelliSense).

1. **פתיחת/יצירת c_cpp_properties.json:**
   - אם הקובץ לא קיים בתיקיית vscode. של הפרויקט:
     - לחצו Ctrl+Shift+P כדי לפתוח את חלונית הפקודות של VS Code.
     - הקלידו C/C++: Edit Configurations (json) ובחרו באפשרות זו.
     - פעולה זו תיצור (אם לא קיים) קובץ **לא ריק** בשם c_cpp_properties.json בתוך התיקייה vscode.
     - במידת הצורך, נפתח את הקובץ c_cpp_properties.json שנוצר על ידי לחיצה כפולה.
2. **הוספת נתיב ה-include של Raylib:**
   - **ודאו שהקובץ c_cpp_properties.json מכיל את ההגדרות הבאות.** החלק החשוב הוא הוספת הנתיב C:/msys64/ucrt64/include למערך includePath.

```
Unset
{
    "configurations": [
        {
            "name": "Win32",
            "includePath": [
                "${workspaceFolder}/**",
                "C:/msys64/ucrt64/include"
            ],
            "defines": [],
            "compilerPath": "C:/msys64/ucrt64/bin/gcc.exe",
            "cStandard": "c11", // מומלץ להשתמש ב-c11 או c17 במקום c99
            "cppStandard": "c++17",
            "intelliSenseMode": "windows-gcc-x64"
        }
    ],
    "version": 4
}
```

הסבר:

- includePath: רשימת התיקיות שבהן VS Code יחפש קבצי header (הקבצים בעלי סיומת .h).

- "${workspaceFolder}/**": כולל את כל קבצי ה-header בתיקיית הפרויקט ובתתי-התיקיות שלה.
- "C:/msys64/ucrt64/include": הנתיב שבו מותקנים קבצי ה-header של ספריות שהותקנו דרך pacman בסביבת UCRT64, כולל raylib.h.
- compilerPath: נתיב מלא לקומפיילר GCC.
- cStandard: תקן שפת סי. c11 או c17 מומלצים על פני c99 הישן יותר.

שלב 3: בדיקת ההתקנה עם תוכנית דוגמה

כדי לוודא שהכל הוגדר כראוי, ניצור תוכנית C פשוטה המשתמשת ב-Raylib. העתיקו את קטע הקוד הבא המכיל פונקציית main לקובץ ה-c. שלכם, עדכנו את ה-PATH לתמונה שנמצאת במחשב שלכם (בשורה המודגשת) והריצו את הקוד.

```
C/C++
/**********************************
* Class: MAGSHIMIM C2            *
* Raylib image display example   *
**********************************/
#include "raylib.h"
#include <stdio.h>

int main(void)
{
    // Initialization
      // You'll need to know the image dimensions or set default window
dimensions.
    // Let's try to load the image first to get its dimensions.
      const char *imagePath = "C:\...changenamehere.jpg"; // Make sure this
path is correct

    // Load image into CPU memory (Image object)
    Image image = LoadImage(imagePath);

    if (image.data == NULL) // Check if image loading failed
    {
        printf("Could not load image: %s\n", imagePath);
        return -1; // Indicate an error
    }

    // Use image dimensions for the window if loaded successfully
    const int screenWidth = image.width;
    const int screenHeight = image.height;

    InitWindow(screenWidth, screenHeight, "Raylib Display Window");

    // Load image from CPU memory (Image) to GPU memory (Texture2D)
    Texture2D texture = LoadTextureFromImage(image);
```

```
    // Unload image from CPU memory (Image) as it's no longer needed
    UnloadImage(image);

    SetTargetFPS(60); // Set our game to run at 60 frames-per-second

    // Main game loop
    while (!WindowShouldClose()) // Detect window close button or ESC key
    {
        // Draw
        BeginDrawing();

        ClearBackground(RAYWHITE); // Clear the background to white

            DrawTexture(texture, 0, 0, WHITE); // Draw the texture at the
top-left corner

         DrawText("Press ESC to close", 10, 10, 20, DARKGRAY); // Optional:
display help text

        EndDrawing();
    }

    // De-Initialization
    UnloadTexture(texture); // Unload texture from GPU memory

    CloseWindow(); // Close window and unload OpenGL context

    return 0;
}
```

אם הכל עובד, אתם אמורים לראות חלון גרפי עם התמונה שלכם!

שימו לב: לפני תחילת העבודה על הפרויקט, הסירו את הקוד שהרצתם להצגת תמונה כך שתוכלו לכתוב פונקציית main בעצמכם. אל תדאגו - הקוד הדרוש להצגת תמונות יסופק לכם על ידינו בקובץ נפרד.