

Színusz függvény ábrázolása 2D-ben

Az alábbi rövid példa egy színusz függvény ábrázolását mutatja.

```
#include <raylib.h>
#include <math.h>

int main() {
    // Initialization
    const int screenWidth = 800;
    const int screenHeight = 450;
    InitWindow(screenWidth, screenHeight, "Sinus Function Plot - raylib");

    SetTargetFPS(60); // Set FPS

    while (!WindowShouldClose()) { // Main game loop
        // Update

        // Draw
        BeginDrawing();

            ClearBackground(RAYWHITE); // Clear the background

            // Draw the axes
            Vector2 origin = { (float)screenWidth/2, (float)screenHeight/2 };

            DrawLine(origin.x, 0, origin.x, screenHeight, BLACK); // Y-axis
            DrawLine(0, origin.y, screenWidth, origin.y, BLACK); // X-axis

            // Draw the sine function
            for(int i = -screenWidth/2; i < screenWidth/2; i++) {
                // Calculating points
                float x1 = (float)i;
                float y1 = sinf(x1 * DEG2RAD) * 100; // Scale the sine wave
                float x2 = x1 + 1;
                float y2 = sinf(x2 * DEG2RAD) * 100; // Scale the sine wave

                // Transform points to screen space
                x1 += origin.x;
                y1 = origin.y - y1; // Invert y1 to match screen coordinates
                x2 += origin.x;
                y2 = origin.y - y2; // Invert y2 to match screen coordinates

                // Draw line segment
                DrawLine(x1, y1, x2, y2, BLUE);
            }
    }
}
```

```
        DrawText("Sinus Function Plot", 10, 10, 20, BLACK); // Title
        DrawText("X-Axis", screenWidth - 50, origin.y + 10, 10, BLACK);
// X-axis label
        DrawText("Y-Axis", origin.x + 10, 10, 10, BLACK); // Y-axis
label

        EndDrawing();
    }

    CloseWindow();

    return 0;
}
```

A 6. és 7. sorban megadjuk a képernyő méretet pixelekben. Ezt a két értéket kapja meg paraméterként az *InitWindow()* függvény.

A 22.-es sorban egy vízszintes vonalat rajzolunk a képernyő közepére. Majd egy függőlegeset ami középen metszi.

A 26. sorban kezdődő ciklus *i* változója szögben számol, a kezdőértéke -400 a vége +399 fok lesz.

Nem pontokban, hanem szakaszokban gondolkodunk, hogy folytonos legyen a kirajzolás, ezért a 30. sorban 1 fokkal jobbra is kiszámoljuk a függvényt értékét. A 29. sorban az *y1* értéke -1 és +1 közzé esik a *sin()* definíciója miatt ezért beszorozzuk 100-al, hogy függőlegesen széthúzzuk.

A 34. sortól azért számoljuk át az értékeket, mert a koordináta rendszer kezdőpontja valójában a bal felső sarokban van és a függőleges tengely fentről lefelé növekszik.

Térbeli ábrázolás

A következő példa egy kétváltozós függvényt **$\text{sinf}(\text{sqrtf}(x*x + y*y))$** térben ábrázol

```
#include <raylib.h>
#include <math.h>

int main() {
    // Initialization
    const int screenWidth = 800;
    const int screenHeight = 800;
    InitWindow(screenWidth, screenHeight, "3D Sinus Function Plot - raylib");

    // Define the camera
    Camera camera = { 0 };
    camera.position = (Vector3){ 20.0f, 20.0f, 20.0f };
}
```

```

camera.target = (Vector3){ 0.0f, 0.0f, 0.0f };
camera.up = (Vector3){ 0.0f, 1.0f, 0.0f };
camera.fovy = 45.0f;

SetTargetFPS(60);

while (!WindowShouldClose()) {
    BeginDrawing();
        ClearBackground(RAYWHITE);
        BeginMode3D(camera);

            for (float y = -8.0f; y < 8.0f; y += 0.2f) {
                for (float x = -8.0f; x < 8.0f; x += 0.2f) {
                    float z = sinf(sqrtf(x*x + y*y)) * 2.0f; //
Amplitude increase for better visualization
                    Vector3 pos = { x, z, y };
                    float colorIntensity = (z + 2.0f) / 4.0f; //
Normalize z value to [0, 1] for color
                    Color color = ColorFromHSV(200.0f * colorIntensity,
0.8f, 0.8f);

                    DrawCubeV(pos, (Vector3){0.2f, 0.1f, 0.2f}, color);
                }
            }
        DrawGrid(20, 1.0f);
        EndMode3D();
        DrawFPS(10, 10);
    EndDrawing();
}
CloseWindow();
return 0;
}

```

Hogyan lehetne z tengely körül forgatni a függvényt?

Ahhoz, hogy a 3D-s sinus felület forogjon a Z tengely körül, frissíteni kell a kamerapozíciót a fő cikluson belül, hogy a kamera mozogjon a felület körül. A kamera mozgatásához polár koordinátákat használunk, és változtatjuk a kamera position vektorát a Z tengely körüli körpályán.

Viszont a számítógépes grafikában az y tengely a felfelé mutató vektor, ezért a kódban az x és z tengelyeket forgatjuk.

```

#include <raylib.h>
#include <math.h>

int main() {
    // Initialization
    const int screenWidth = 800;
    const int screenHeight = 800;
    InitWindow(screenWidth, screenHeight, "3D Sinus Function Rotating -

```

```
raylib");

// Define the camera
Camera camera = { 0 };
camera.position = (Vector3){ 30.0f, 20.0f, 3.0f };
camera.target = (Vector3){ 0.0f, 0.0f, 0.0f };
camera.up = (Vector3){ 0.0f, 1.0f, 0.0f };
camera.fovy = 45.0f;

SetTargetFPS(60);
float angle = 0.0f;

while (!WindowShouldClose()) {
    // Update camera position
    angle += 0.01f; // Increment angle
    camera.position.x = cosf(angle) * 30.0f;
    camera.position.z = sinf(angle) * 30.0f;

    BeginDrawing();
        ClearBackground(RAYWHITE);
        BeginMode3D(camera);

            for (float y = -8.0f; y < 8.0f; y += 0.2f) {
                for (float x = -8.0f; x < 8.0f; x += 0.2f) {
                    float z = sinf(sqrtf(x*x + y*y)) * 2.0f; //
Amplitude increase for better visualization
                    Vector3 pos = { x, z, y };
                    float colorIntensity = (z + 2.0f) / 4.0f; //
Normalize z value to [0, 1] for color
                    Color color = ColorFromHSV(200.0f * colorIntensity,
0.8f, 0.8f);
                    DrawCubeV(pos, (Vector3){0.2f, 0.1f, 0.2f}, color);
                }
            }
        DrawGrid(20, 1.0f);
        EndMode3D();
        DrawFPS(10, 10);
    EndDrawing();
}
CloseWindow();
return 0;
}
```

From:

<https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link:

https://edu.iit.uni-miskolc.hu/muszaki_informatika:raylib_vis_functions?rev=1708456809

Last update: **2024/02/20 19:20**

