Geometriai modellezés
Féléves egyéni feladat

Készítette: Dobra Gábor

Neptunkód: XQBTIW

Transzformáció

A geometriai transzformáció geometriai objektumok között létesitett megfeleltetés, reláció. Fogalmai szemléletes konkrétumokból származtathatók, de a tudomány fejlődése során absztrakttá váltak és korábban nem tapasztalt jellemzőkkel egészültek ki. Esetenként geometriai alakzatok, máskor a sík/tér minden pontjának áthelyezéseként-átalakításaként értelmezzük. Ugyanígy váltakozva a sík önmagára vagy egy másik síkra való transzformálásáról beszélünk. Ugyancsak transzformáció a térbeli alakzatok síkbeli szemléltetése: ábrázolása, de pont-pont megfeleltetést valósítunk meg a Földet síkbanábrázoló térképeken is.

Története:

Eukleidész az Elemekben csak két alakzat egybevágóságával és hasonlóságával, mint az összehasonlítás egyik attribútumával foglalkozik anélkül, hogy ezeket megnevezné. Csupán úgy szólnak a tételei, hogy "bizonyos méretek egyezése esetén más méretek is egyelők". Csupán úgy szólnak a tételei, hogy "bizonyos méretek egyezése esetén más méretek is egyelők". Az Elemek szellemében keletkezett ókori és középkori munkákban csupán nyomokban jelenik meg a megfeleltetés, s akkor is inkább olyan feladatokban, hogy "szerkesszük meg egy adott alakzat olyan képét ..." és az olyan kikötésében fogalmazódik meg az egybevágóság/hasonlóság kritériuma. Az Elemek szemlélete a szintetikus geometriát tárgyaló tankönyvek, monográfiák lapjain még a XX. sz-ban is felfedezhető. Az elnevezés eredete bizonytalan és nem is találó, hiszen mint a latin ***transformare*** ige származéka átalakítást jelent, de sok transzformáció éppenhogy nem változtatja meg az alakzatok formáját.

1872, ekkor hangzott el ***Felix Klein*** (1849–1925) német matematikus nevezetes előadása, amelyre a matematika története az Erlangeni Program néven utal. Klein ebben az összefoglalójában hívta fel a figyelmet, hogy a geometriai transzformációkat vizsgálhatjuk aszerint is, hogy egyes alakzatoknak milyen tulajdonságait örökítik, más szóhasználattal, hogy a transzformáció során melyik tulajdonság változatlan, invariáns.

A következőkben egy háromszög geometriai transzformációját fogom bemutatni, amelyet a vizsgálat során, különböző értékek megadásával figyelhetünk meg.

Header file- ok implementálása, a program müködéséhez szükségesek:

```
#define _CRT_SECURE_NO_WARNINGS
#include <GL/glew.h>
#include <SDL3/SDL.h>
#include <stdio.h>
#include <cmath>
```

A háromszöget kirajzoló függvény, amely tartalmaz három két elemű tömböt, a háromszög x, y koórdinátáohoz:

```
void haromszogRajz(float A[2], float B[2], float C[2]){
glBegin(GL_LINE_LOOP);
          glVertex2f(A[0], A[1]);
          glVertex2f(B[0], B[1]);
          glVertex2f(C[0], C[1]);
glEnd();
}
int main(int argc, char* argv[])
{
```

Különböző alap paraméterek megadása, az ablak kirajzólásának, méretének:

```
    SDL_Window* window;
    SDL_GLContext gl_context;
    int error_code;
    int width = 800;
    int height = 600;
```

Felvesszük a változókat, és adunk nekük kezdő értéket:
(ox-oy, forgatás középpontja)

```
    float ox = 420;
    float oy = 310;
```

R: forgatás szöge:

```
float R = -90;
```

A tx, ty, eltolás mértéke:

```
float tx = 0;
float ty = 0;
```

Az s, növelés szorzója:

```
float s = 1;
```

Felhasználótól kérjük be a transzformációs paramétereket:

```
printf("Adja meg a forgatás középpontjának x koordinátáját: ");
scanf("%f", &ox);
printf("Adja meg a forgatás középpontjának y koordinátáját: ");
scanf("%f", &oy);
printf("Adja meg a forgatás szögét fokokban: ");
scanf("%f", &R);
printf("Adja meg, hányszorosára nagyítsuk a háromszögeket: ");
scanf("%f", &s);
printf("Adja meg az eltolás x koordinátáját: ");
scanf("%f", &tx);
printf("Adja meg az eltolás y koordinátáját: ");
scanf("%f", &ty);
```

Létrehozunk egy ablakot, ”Eltolás” néven:

```
SDL_WindowFlags flags = SDL_WINDOW_OPENGL | SDL_WINDOW_RESIZABLE;
window = SDL_CreateWindow("An SDL3 window", 840, 680, SDL_WINDOW_OPENGL);
    if (window == NULL) {
        printf("[ERROR] Unable to create the application window!\n");
    }
}
```

OpenGL context létrehozása:

```
gl_context = SDL_GL_CreateContext(window);
    if (gl_context == NULL) {
        printf("[ERROR] Unable to create the OpenGL context!\n");
    }
```

Alapbeállitások: fekete kitöltőszín, vetitési sik és nézőpont definialása:

```
glClearColor(0.5, 0.5, 0.5, 1); // háttérszín szürke
glMatrixMode(GL_PROJECTION);
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
glOrtho(0, width, height, 0, -200, 200); // ortografikus vetítés
glViewport(0, 0, width, height);
```

Háromszög pontjainak létrehozása, amelynek minden pontja egy két elemű tömb:

```
float A[] = {390, 300};
float B[] = {450, 300};
float C[] = {420, 320};
```

Az első háromszög kirajzolása:

Eltoljuk a megadott pontba, megnöveljük x és y mentén kétszeresre, visszahúzzuk az eredeti helyére:

```
glPushMatrix();
   glTranslatef(ox, oy, 0);
   glScalef(s, s, 0);
   glTranslatef(-ox, -oy, 0);
   haromszogRajz(A, B, C);
glPopMatrix();
```

Második háromszög kirajzolása:

Eltoljuk a megadott pontba, annak a középpontja szerint elforgatjuk a megadott R szöggel a z tengely körül ezután megnöveljük kétszeresére, majd a koordinátarendszert visszahúzzuk az eredeti helyére, majd az egészet eltoljuk a 100-100 pontba.

```
    glPushMatrix();
         glTranslatef(ox, oy, 0);
         glRotatef(R, 0, 0, 1);
         glScalef(s, s, 0);
         glTranslatef(-ox, -oy, 0);
    glTranslatef(tx, ty, 0);
```

Kirajzoljuk a háromszöget, majd visszavesszük az eredeti transzformációs mátrixot:

```
     glColor3f(1, 0, 1);
         haromszogRajz(A, B, C);
     glPopMatrix();
```

Megjelenítés:

```
  glFlush();
    SDL_GL_SwapWindow(window);
```

Program szüneteltetése bezárás előtt:

```
  system("PAUSE");
```

Erőforrások lezárása:

```
    SDL_GL_DestroyContext(gl_context);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

Forráskód:

```
#define _CRT_SECURE_NO_WARNINGS
#include <GL/glew.h>
#include <SDL3/SDL.h>
#include <stdio.h>
#include <cmath>

void haromszogRajz(float A[2], float B[2], float C[2]) {
    glBegin(GL_LINE_LOOP);
    glVertex2f(A[0], A[1]);
    glVertex2f(B[0], B[1]);
    glVertex2f(C[0], C[1]);
    glEnd();
}

int main(int argc, char* argv[]) {
    SDL_Window* window;
    SDL_GLContext gl_context;
    int error_code;
    int width = 800;
    int height = 600;

    float ox = 420;
    float oy = 310;
    float R = -90;
    float tx = 0;
    float ty = 0;
    float s = 1;

    printf("Adja meg a forgatas kozeppontjanak x koordinatajat: ");
    scanf("%f", &ox);
    printf("Adja meg a forgatas kozeppontjanak y koordinatajat:");
    scanf("%f", &oy);
    printf("Adja meg a forgatas szoget fokokban: ");
    scanf("%f", &R);
    printf("Adja meg, hanyszorosara nagyitsuk a haromszogeket: ");
    scanf("%f", &s);
    printf("Adja meg az eltolas x koordinatajat: ");
    scanf("%f", &tx);
    printf("Adja meg az eltolas y koordinatajat: ");
    scanf("%f", &ty);

    SDL_WindowFlags flags = SDL_WINDOW_OPENGL | SDL_WINDOW_RESIZABLE;
         window = SDL_CreateWindow("An SDL3 window",
     840,680,SDL_WINDOW_OPENGL
    );
```

```
    SDL_Window* SDL_CreateWindow(const char* title, int w, int h,
    Uint32 flags);
     if (window == NULL) {
        printf("[ERROR] Unable to create the application window!\n");
    }

    gl_context = SDL_GL_CreateContext(window);
    if (gl_context == NULL) {
        printf("[ERROR] Unable to create the OpenGL context!\n");
    }

    glClearColor(0.5, 0.5, 0.5, 1);
    glMatrixMode(GL_PROJECTION);
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glOrtho(0, width, height, 0, -200, 200);
    glViewport(0, 0, width, height);

    float A[] = { 390, 300 };
    float B[] = { 450, 300 };
    float C[] = { 420, 320 };

    glPushMatrix();
    glTranslatef(ox, oy, 0);
    glScalef(s, s, 0);
    glTranslatef(-ox, -oy, 0);
    haromszogRajz(A, B, C);
    glPopMatrix();

    glPushMatrix();
    glTranslatef(ox, oy, 0);
    glRotatef(R, 0, 0, 1);
    glScalef(s, s, 0);
    glTranslatef(-ox, -oy, 0);
    glTranslatef(tx, ty, 0);

    glColor3f(1, 0, 1);
    haromszogRajz(A, B, C);
    glPopMatrix();

    glFlush();
    SDL_GL_SwapWindow(window);

    system("PAUSE");
    SDL_GL_DestroyContext(gl_context);
    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
}
```