02_SDLEvents

Forrás: http://cg.elte.hu/~bsc_cg/gyak/01/02_SDLEvents.zip

Az előző feladat ismereteit felhasználva már képesek vagyunk az SDL segítségével megnyitni egy ablakot és rajzolni rá. A következő lépés, hogy a programot interaktívvá tegyük, azaz billentyűzettel és egérrel vezéreljük.

Ehhez lényegében a 01_HelloSDL programot írjuk tovább:

```
int main( int argc, char* args[] )
{
    atexit( exitProgram );
    //
    // 1. lépés: inicializáljuk az SDL-t
    //

    // a grafikus alrendszert kapcsoljuk csak be, ha gond van, akkor jelezzük és lépjün ki
    if ( SDL_Init( SDL_INIT_VIDEO ) == -1 )
    {
        // irjuk ki a hibat es terminaljon a program
        std::cout << "[SDL indítása]Hiba az SDL inicializálása közben: " << SDL_GetError() << std::endl;
        return 1;
    }

    //
    // 2. lépés: hozzuk létre az ablakot, amire rajzolni fogunk
    //

    SDL_Window *win = 0;
    win = SDL_CreateWindow( "Hello SDL!"                    // az ablak feiléce
```

Ahhoz, hogy a program az egér és a billentyűzet eseményeit figyelhesse, szükség van egy eseményfigyelőre. Az SDL-ben már léteznek az ehhez a feladathoz készített objektumok. Az „esemény sor” egy adatszerkezet, ami a háttérben tárolja az `SDL_Event` típusú eseményeket. A sor elemeit az `SDL_PollEvent(&ev)` hívással tudjuk elérni. Ez az `ev` változóba teszi az első eseményt a sorból. Ebben a változóban lesznek eltárolva az esemény adatai, amik alapján később lekezelhetjük.

Egy végtelen ciklusba helyezzük a kirajzolást, hiszen nem szeretnénk, hogy a programunk azonnal leálljon, mert így nem figyelné az eseményeket.
Az `SDL_PollEvent(&ev)` hívást szintén egy ilyen ciklusba tesszük, hogy minden kirajzolásnál az összes eseményt kivegyük a sorból.

```
// véget kell-e érjen a program futása?
bool quit = false;
// feldolgozandó üzenet ide kerül
SDL_Event ev;
// egér X és Y koordinátái
Sint32 mouseX = 0, mouseY = 0;

while (!quit)
{
    // amíg van feldolgozandó üzenet dolgozzuk fel mindet:
    while ( SDL_PollEvent(&ev) )
    {
        switch (ev.type)
        {
        case SDL_QUIT:
            quit = true;
            break;
        case SDL_KEYDOWN:
            if ( ev.key.keysym.sym == SDLK_ESCAPE )
                quit = true;
            break;
        case SDL_MOUSEMOTION:
            mouseX = ev.motion.x;
            mouseY = ev.motion.y;
            break;
```

Az `ev.type` tárolja az aktuális esemény típusát, mint egérmozgás, billentyűlenyomás stb… Ez alapján elkülöníthetjük ezeket az eseményeket. Billentyűkezelésnél az eseményt küldő billentyűt is lekérdezhetjük. Ez az `ev.key.keysym.sym`-ben van tárolva.

Miután a változóinkat megváltoztattuk, az aktuális értékeikkel megkezdődhet a kirajzolás. Ez változatlan az előző programhoz képest:

```
// töröljük a hátteret fehérre
SDL_SetRenderDrawColor(ren, 255, 255, 255, 255);
SDL_RenderClear(ren);

// aktuális rajzolási szín legyen zöld és rajzoljunk ki egy vonalat
SDL_SetRenderDrawColor( ren,    // renderer címe, aminek a rajzolási színét be akarjuk állítani
                        0,      // piros
                        255,    // zöld
                        0,      // kék
                        255);   // átlátszatlanság

SDL_RenderDrawLine( ren,    // renderer címe, ahová vonalat akarunk rajzolni
                    0, 0, // vonal kezdőpontjának (x,y) koordinátái
                    mouseX, mouseY);// vonal végpontjának (x,y) koordinátái

// definiáljunk egy (mouseX, mouseY) középpontó, tengelyekkel párhuzamos oldalú
// 20x20-as négyzetet:
SDL_Rect cursor_rect;
cursor_rect.x = mouseX - 10;
cursor_rect.y = mouseY - 10;
cursor_rect.w = 20;
cursor_rect.h = 20;
// legyen a kitöltési szín piros
SDL_SetRenderDrawColor( ren, 255, 0, 0, 255 );
```

A kirajzolás után újrakezdődik a folyamat, és addig folytatódik, amíg be nem zárjuk a programot.

Ha tudni szeretnéd, hogy milyen esemény típusok vannak az SDL-ben, vagy hogy hogyan kell egy betű SDL-kódját leírni, a Visual Studioban egy jobbegér nyomással megnézheted a headerfileokat, ahol könnyen megtalálhatóak ezek az adatok.