SDL – Multimédiás programozói könyvtár (2. rész)

Az előző részben próbáltunk átfogó képet kapni az SDL rendszerről, valamint belekezdtünk a video funkciók igen gazdag családjának használatába. A mostani cikkben még mindig a képi megjelenítéssel fogunk foglalkozni. Megnézzük, hogyan is lehet képeket betölteni és menteni SDL használatával, de fel fog bukkanni pár új eseménykezeléssel kapcsolatos újdonság is.

Amit szeretnénk

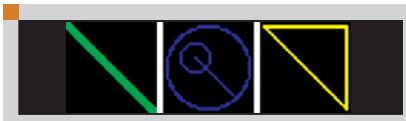
Lássuk mit is szeretnénk ez alkalommal alkotni. Készítsünk egy olyan *SDL* alkalmazást, melyben az egér segítségével rajzolni tudunk, lehetőleg többféle ecset segítségével, lehessen törölni a rajzfelületet, valamit elmenti annak tartalmát. Oldjuk meg azt is, hogy az alkalmazás tetszés szerint teljes képernyőn illetve ablakban fusson.

Ahogyan szeretnénk

Maradunk a *C++* és *SDL* ötvözeténél. Az előző számban már láthattuk, hogyan is kell az *SDL* rendszert inicializálni, ezért erre már nem térünk ki különösebben. Mindenképpen szükségünk lesz eseménykezelésre is, amiből szintén kaptunk már egy nagyon kicsi ízelítőt. Most az egérkezelés fog előtérbe kerülni, de nem felejtjük el a billentyűzetet sem.

Előkészületek

Mindenek előtt készítsük el az ecseteket, valamilyen rajzprogram segítségével. A kép legyen 32x32 képpont, fekete háttérrel, és rajzoljunk rá valamit, ami szerintünk megfelel majd az ecsetünk mintázatának. Lényeges, hogy az ábrát *BMP* formában mentsük (*1. ábra*).



1. ábra Az ecsetek

Kezdhetjük

Most, hogy minden forrásunk megvan a programozáshoz, kezdjük el a munkánk nagyobbik és komolyabb részét. Először is szükségünk lesz egy rajzfelületre, ez esetben 800x600-as felbontás már igen kényelmes méretnek bizonyul. Szükség lesz még ezenkívül még egy `SDL_Surface`-re, amire az ecseteket fogjuk tölteni. Rajzoláskor ezt a felületet fogjuk „átbillenteni” a rajzfelületünkre, ott ahol éppen az egerünk jár. Mivel a rajzunkat úgy készítettük, hogy a háttér fekete, ezért erre állítunk be egy színkulcsot, ezen a felületen, ahol az aktuális ecset ábrája van. Ez azért lesz jó mert mi csak az ábrával szeretnénk rajzolni, a háttér számunkra felesleges, tehát a fekete szín a mi esetünkben átlátszó kell hogy legyen. Ezt valósítjuk meg az `SDL_SetColorKey` függvénnyel. Első paramétere, annak a felületnek a mutatója melyre a kulcsot szeretnénk „ráhúzni”. Második paraméterként flageket vár.
A mostani flagek:
`SDL_SRCCOLORKEY,SDL_RLEACCEL`.
Az első tájékoztatja az *SDL*-t, hogy ez a felület forrásként fog szolgálni az átbillentés (egyik felület másolása a másikra) során. A második flag, pedig a billentés gyorsításáért felelős. Harmadszor pedig azt a színt adjuk meg melyet nem szeretnénk látni az `SDL_MapRGB` függvénnyel (erről már volt szó az előző cikkben). A kódunk tehát az *1. Listában* látható módon alakul. A `brush1,brush2,brush3` változók tárolják az egyes ecsetekhez tartozó elérési utakat. A `saveto` változó pedig annak a fájlnak a nevét melybe, majd menteni fogjuk a rajzfelületet. Kezdetnek töltsük be a `brush_surface`-re az

1. Lista – Kezdjük a rajzoló alkalmazást – felületek, inicializálás

```
#include <iostream>
#include "SDL.h"
int main()
{
    SDL_Surface *sdl_surface;
    SDL_Surface
    ➥*brush_surface;
    SDL_Event sdl_event;
    bool main_loop_exit =
    ➥false;
    char *saveto =
    ➥"sdl_surface.bmp";
    char *brush1  =
    ➥"brush1.bmp";
    char *brush2  =
    ➥"brush2.bmp";
    char *brush3  =
    ➥"brush3.bmp";

   // inicializációs rész
   brush_surface =
   ➥SDL_LoadBMP(brush1);
    SDL_SetColorKey
    ➥(brush_surface,
    ➥SDL_SRCCOLORKEY |
    ➥SDL_RLEACCEL,
    ➥SDL_MapRGB
    ➥(brush_surface->
    ➥format,0,0,0));
...
```

2. lista – Eseménykezelő ciklusunk

```
...
 while (!main_loop_exit)
 {
   while (SDL_PollEvent
   ➥(&sdl_event))
       {
           switch
           ➥(sdl_event.type)
           {
       case SDL_KEYDOWN:
       ...
       case SDL_MOUSEBUTTON:
       ...
       case SDL_MOUSEMOTION:
       ...
       default: break;
     }
   }
 }
```

3. lista – Billentyűk azonosítása

```
...
if (sdl_event.key.keysym.sym
➥== SDLK_1)
                    {
          brush_surface =
          ➥SDL_LoadBMP
          ➥(brush1);
          SDL_SetColorKey
(brush_surface,
➥SDL_SRCCOLORKEY |
➥SDL_RLEACCEL,
➥SDL_MapRGB(brush_surface->
➥format,0,0,0));
                    }
if (sdl_event.key.keysym.sym
➥== SDLK_2)
                    {
    brush_surface =
    ➥SDL_LoadBMP(brush2);
    ...
                    }
...
```

4. lista – Egérmozgás

```
...
case SDL_MOUSEMOTION:
       x =
       ➥sdl_event.button.x;
       y =
       ➥sdl_event.button.y;
       if (SDL_GetMouseState
       ➥(NULL,NULL) &
       ➥SDL_BUTTON(1))
       {
          paint(x, y,
          ➥brush_surface,
          ➥sdl_surface);
           SDL_UpdateRect
           ➥(sdl_surface, x,
           ➥y,32,32);
        }
        break;
...
```


`brush1.bmp`-t. Az *SDL* a *BMP* képek betöltésére rendelkezik egy külön `SDL_LoadBMP` nevű eljárással, mentésére pedig analóg módon az `SDL_SaveBMP` használható.

Események

Az alkalmazást végül is inicializáltuk. Most a nagy eseménykezelő ciklus fog következni, mely hasonló az elő cikkben szerepelt példához, csak most egy kicsit bővíteni fogjuk. Oldjuk meg azt, hogy az alkalmazásunk csak az *ESC* gomb megnyomására lépjen ki. Most a *2. Listában* látható módon így fog alakulni a ciklusunk.
Az `SDL_KEYDOWN` ágon, egyértelműen a billentyűzetet fogjuk kezelni, az `SDL_MOUSEBUTTON` ág az egérgombokat fogja figyelni, amíg az `SDL_MOUSEMOTION` az egér mozgására fog koncentrálni. Lássuk hát, hogyan fog kilépni a program *ESC* billentyűre. A ciklusunk feltételét a `main_loop_exit` változó képviseli. Elég ennyi a megoldáshoz az `SDL_KEYDOWN` ágon:

```
main_loop_exit =
➥(sdl_event.key.keysym.sym ==
➥SDLK_ESCAPE)
```

A lenyomott billentyűket az `esemeny.key.keysym.sym` változó alatt érhetjük el. Minden billentyűnek van egy konstansa az *SDL*-ben, például `SDLK_ESCAPE`, `SDLK_a`, `SDL_b`, `SDL_1`,... Az `SDL_Key` címszó alatt megtalálható az ide vonatkozó táblázat a dokumentációban. A teljes képernyő és ablakos mód közötti váltás is ezen fog alapulni:

```
if (sdl_event.key.keysym.sym
➥== SDLK_f)
➥SDL_WM_ToggleFullScreen
➥(sdl_surface)
```

Az `SDL_WM_ToggleFullScreen` eljárás az adott felületre váltogatja a teljes képernyős és ablakos módokat. Legyen például az 1-es billentyű az *brush1.bmp*-t használó ecsetünk és így rendre a többi (*3. Lista*). Elvárjuk azt is, hogy a képernyő letörölhető legyen. Ezt mondjuk társítsuk a *D* billentyűhöz. Az `SDL_FillRect` eljárás segítségével az első paraméterként megadott felületet a következő három paraméterben megadott RGB értékekkel tölthetjük fel. Töröljük hát a „rajzlapunkat”:

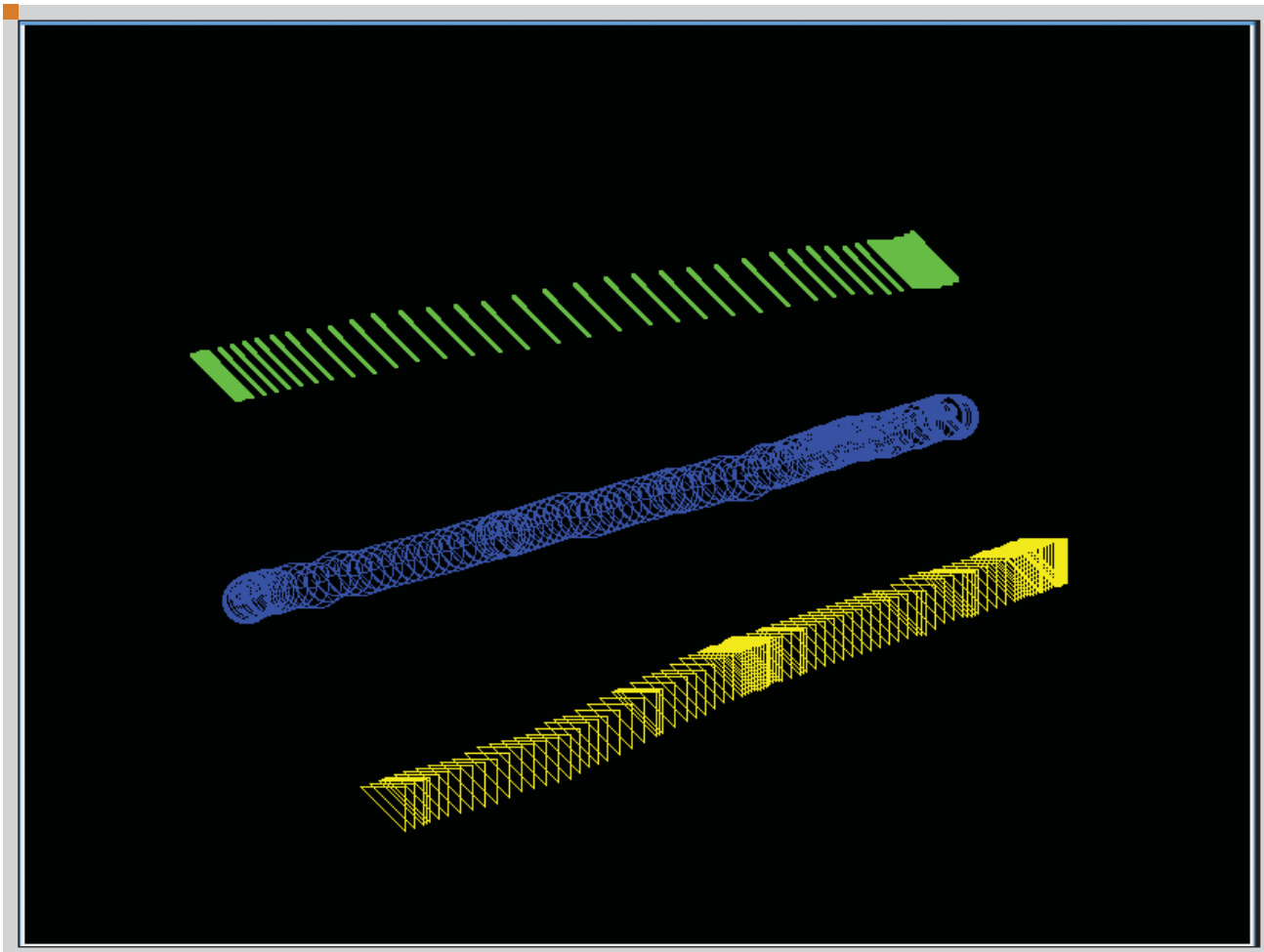
```
SDL_FillRect(sdl_surface,0,0,0)
```

Ne felejtsük el a felületet frissíteni az `SDL_UpdateRect`-el!
Maradt még a kép elmentése. Alkalmazzuk erre az *S* gombot. Az `SDL_SaveBMP` egy `SDL_Surface`-t és egy fájlnevet vár paraméterként, amibe a felület tartamát lementheti:

```
SDL_SaveBMP(sdl_surface,saveto)
```

Egér

Térjünk most át az egérre. Az `SDL_MOUSEMOTION` esemény aktiválódik, ha az egeret megmozdítjuk. Közben már csak figyelnünk kell, hogy le van-e nyomva a bal egérgomb vagy sem. Rajzolunk ha igen (*4. Lista*). Hogy hol is jár éppen az egér azt az `sdl_event.button.x`, `sdl_event.button.y` változók fogják nekünk megmondani. A feltételben vizsgáljuk, hogy le van-e nyomva a bal egérgomb ha igen, akkor rajzolunk. A `paint` eljárás fogja az ecsetet tartalmazó felületet átbillenteni a „rajzlapra” (*5. Lista*).
Ami új az az `SDL_BlitSurface`. Az első két paraméter egy felület és egy `SDL_Rect` típus. Az `SDL_Rect` egy négyszögterületet tartalmaz, *x,y* a koordináták, w a szélesség, h a magasság. Ha éppen ez `NULL` mutató, akkor



2. ábra Az egyes ecsetek mintái

az egész felület másolásra kerül. A második két paraméter is egy felület-rect páros. Ez már a célfelület és a cél terület a felületen. Itt a `NULL` mutató a `rect` érték helyén, azt eredményezi, hogy a forrás felület a célfelület 0,0 koordinátájában fog megjelenni.

5. Lista – Az ecset kirajzolása megadott pozícióba

```
void paint(Sint16 x,Sint16 y,
➥SDL_Surface
➥*src,SDL_Surface *dst)
{
    SDL_Rect *dst_rect = new
    ➥SDL_Rect;
    dst_rect->x = x;
    dst_rect->y = y;
    SDL_BlitSurface
    ➥(src,NULL,dst,dst_rect);
}
```

Mi most csak az ***x,y*** koordinátákat adtuk meg, mivel az `SDL_BlitSurface` csak ezeket veszi figyelembe a struktúrából. Tehát az ecsetet tartalmazó kép a rajzlap x,y koordinátájára fog kerülni. A koordinátákat pedig már az egér eseménykezelője fogja átadni. A `paint` után természetesen frissíteni kell a „rajzlapot”. Rendben is volnánk, de ez a kezelő csak akkor rajzol, ha mozgatjuk az egeret. Mi van ha csak egy kattintásra szeretnénk rajzolni. Erre fogjuk alkalmazni az `SDL_MOUSEBUTTON` ágat. Ide ugyanazt a kódot írjuk, amit az `SDL_MOUSEMOTION` ághoz, csak a feltételt elhagyjuk.
Ha minden rendben ment, most van egy alkalmazásunk, mely megfelel a cikk elején említett elvárásoknak. Megismerkedtünk közelebbről a billentyűzet és az egér eseménykezelésével, valamit láttuk, hogyan kell betölteni illetve menteni egy ***BMP*** képet. Alkalmaztuk az `SDL_BlitSurface`-t, ami alapvető minden grafikus ***SDL*** alkalmazásban, segítségével másolhatunk egy felületet a másikra. Láttunk példát színkulcshasználatára, mellyel átlátszóvá tehetünk egy bizonyos színt egy felületen.
A következő részben az ***audio*** eszközökkel fogunk ismerkedni, valamit a ***CD-ROM*** kezelésbe is fogunk kicsit vájkálni, valamit hátra van még az ***időzítő*** funkciók használata. Remélhetőleg hasznos információkkal szolgált a sorozat ezen része is mindazoknak, akik most kezdenek érdeklődni az ***SDL*** iránt. Jövő hónapban folytatjuk!

Radics Péter
(peter.radics@gmail.com)
Az ELTE-n tanulok programtervező matematikus szakon. Hobbim a kosárlabda, autóvezetés, web-design, programozás. Főleg webes alkalmazások fejlesztése érdekel. 4 éve megrögzött Linux felhasználó vagyok.