LinuxFocus article number 288
http://linuxfocus.org

Een scroller implementeren in SDL graphics

door Leonardo Giordani
<leo.giordani(at)libero.it>

Over de auteur:

Ik heb juist mijn diploma ontvangen van de faculteit van Telecommunicatie Engineering in Politecnico in Milaan. Geïnteresseerd in programmeren (meestal Assembly en C/C++). Werkt sinds 1999 bijna alleen maar met Linux/Unix.

Vertaald naar het Nederlands door:
Guus Snijders
<ghs(at)linuxfocus.org>

Kort:

Deze serie artikelen is om de lezer te introduceren in de wereld van multimedia producties, ook wel bekend als "demos". Het hele Internet staat vol met informatie over dit onderwerp, maar slechts weinig mensen schrijven dergelijke schoonheden voor Linux: het doel is om de theorie te beschrijven van dergelijke graphische en audio effecten en de implementatie hiervan met behulp van de SDL library (bibliotheek). Verdere informatie kan gevonden worden op

- www.libsdl.org: het lezen van de code van open source demos en spellen is de beste manier om nieuwe oplossingen te vinden.
- www.lnxscene.org: een nieuwe site, maar kan een aardige plaats zijn om kennis uit te wisselen; je kunt me hier (soms) vinden als "muaddib".

Nodig voor het begrijpen van het artikel is:

- Basiskennis van C (syntax, loops, libraries)
- Basiskennis van SDL library (basis functies, initializeren) --> www.libsdl.org

De scroller

Heel veel dank aan Sam Lantinga voor de SDL library.

Een scroller is een deel van demo dat een bewegende zin op het beeld schrijft: het is een van de basis effecten die je kunt vinden in een multimedia productie en voegt een beetje dynamica toe aan de tekst

die je wilt laten zien aan de gebruiker. In dit artikel zullen zien hoe je een eenvoudige scroller kunt maken die de tekst van rechts naar links beweegt.

Het idee van een scroller is om een deel van het scherm pixel voor pixel naar links te kopiëren (of een andere richting). Het uitvoeren van deze operatie met een goede snelheid geeft je een illusie van een continue beweging, en dat is alles.

De basis theory is niet erg complex; laten we eens kijken hoe dit alles te vertalen is in termen van code: we zullen zo nu en dan refereren aan het concept van oppervlakte, bekend bij een ieder met ervaring met SDL programmering. Als we werken met de SDL library moeten we altijd in gedachten houden dat gebruik kunnen maken van de krachtige

`SDL_BlitSurface()` functie, welke ons toestaat een deel van een `SDL_Surface` te kopiëren, geïdentificeert door een `SDL_Rect` op een ander `SDL_Surface` geïdentificeert door een andere `SDL_Rect` .

Stel je bijvoorbeeld voor dat we twee oppervlaktes en twee rechthoeken hebben gedefiniëerd en geïnitialiseerd.

```
#define WIDTH 800
#define HEIGHT 600

SDL_Surface *Src;
SDL_Surface *Dst;

Src = SDL_CreateRGBSurface(SDL_HWSURFACE, WIDTH, HEIGHT, 8,
       r_mask, g_mask, b_mask, a_mask);
Dst = SDL_CreateRGBSurface(SDL_HWSURFACE, WIDTH, HEIGHT, 8,
       r_mask, g_mask, b_mask, a_mask);

SDL_Rect BigArea = {0, 0, (WIDTH / 2), HEIGHT};
SDL_Rect SmallArea = {(WIDTH / 2) + 1, 0, (WIDTH / 2), (HEIGHT / 2)};
```

waar we er vanuit gaan dat het kleuren masker (color mask) al geïnitialiseerd is. Het kopiëren van de twee volledige oppervlaktes impliceert een minimale inspanning

```
SDL_BlitSurface(Src, 0, Dst, 0);
```

merk op dat de doel oppervlakte alleen de originele coördinaten van de rechthoek worden gebruikt en niet de dimensie: dit betekend dat de operatie

```
SDL_BlitSurface(Src, &BigArea, Dst, &SmallArea);
```

perfect legaal is en de linker helft van de `Src` oppervlakte kopiëert naar de rechterhelft van de `Dst` oppervlakte.

Het uitvoeren van de scroll van een oppervlakte zou nu geen mysterie meer moeten zijn: het is voldoende om een deel van de rechthoek te kopiëren in een verschoven rechthoek op dezelfde oppervlakte; uiteraard dient deze code plaats te vinden in een loop, waardoor het resulterende programma wat complexer wordt, maar het basis concept is simpel. Bij iedere tik van de loop gebruiken we twee rechthoeken, de tweede verschoven ten op zichte van de eerste, in de richting van de scroll, en we kopiëren de oppervlakte op zichzelf van de eerste naar de tweede rechthoek.

```
SDL_Surface *temp;

SDL_Rect From = {1, 0, WIDTH, HEIGHT};
SDL_Rect First = {0, 0, 1, HEIGHT};
SDL_Rect Last = {WIDTH-1, 0, 1, HEIGHT};

temp = SDL_CreateRGBSurface(SDL_HWSURFACE, 1, HEIGHT, 8,
       r_mask, g_mask, b_mask, a_mask);

while(1){
  SDL_BlitSurface(Src, &First, temp, 0);
  SDL_BlitSurface(Src, &From, Src, 0);
  SDL_BlitSurface(temp, &First, Src, &Last);
  SDL_BlitSurface(Src, 0, Screen, 0);
}
```

Zoals je kunt zien, is het niet afdoende om de oppervlakte naar links te laten scrollen: we moeten aan de rechterkant de pixels die het scherm hebben verlaten, invoegen, anders zal de scrollende oppervlakte delen van de laatste kolom laten staan, en zo een ’slepend’ effect veroorzaken. we worden verondersteld reeds een oppervlakte, gelinkt aan het scherm, te hebben. Laten we nu eens kijken naar een compleet programma dat een 480x200 venster opent en de continue scroll van een afbeelding uitvoerd.

```
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"

#define SCREEN_WIDTH 480
#define SCREEN_HEIGHT 200

#if SDL_BYTEORDER == SDL_BIG_ENDIAN
static const Uint32 r_mask = 0xFF000000;
static const Uint32 g_mask = 0x00FF0000;
static const Uint32 b_mask = 0x0000FF00;
static const Uint32 a_mask = 0x000000FF;
#else
static const Uint32 r_mask = 0x000000FF;
static const Uint32 g_mask = 0x0000FF00;
static const Uint32 b_mask = 0x00FF0000;
static const Uint32 a_mask = 0xFF000000;
#endif
```

Deze set definities is standaard in (bijna) elke multimedia productie.

```
static SDL_Surface* img_surface;
static SDL_Surface* scroll_surface;
static SDL_Surface* temp_surface;
```

Dit zijn de drie oppervlakken die zullen gebruiken: `img_surface` zal een afbeelding uit een bestand bevatten, `scroll_surface` de verschoven afbeelding en `temp_surface` de pixels om rechts weer in te voegen.

```
static const SDL_VideoInfo* info = 0;
SDL_Surface* screen;
```

Een `SDL_VideoInfo` structuur bevat informatie over video hardware, terwijl het `screen` oppvlak naar het echte scherm zal verwijzen.

```
int init_video()
```

```
{
  if( SDL_Init( SDL_INIT_VIDEO) < 0 )
    {
      fprintf( stderr, "Video initialization failed: %s\n",
               SDL_GetError( ) );
      return 0;
    }

  info = SDL_GetVideoInfo( );

  if( !info ) {
    fprintf( stderr, "Video query failed: %s\n",
             SDL_GetError( ) );
    return 0;
  }

  return 1;
}

int set_video( Uint16 width, Uint16 height, int bpp, int flags)
{
  if (init_video())
    {
      if((screen = SDL_SetVideoMode(width,height,bpp,flags))==0)
        {
          fprintf( stderr, "Video mode set failed: %s\n",
                   SDL_GetError( ) );
          return 0;
        }
    }
  return 1;
}
```

De `init_video()` functie initialiseert het SDL video subsyteem en vult de `info` structuur op. De `set_video()` functie probeert een gegeven video mode (dimensies en kleur diepte) op te zetten

```
void quit( int code )
{
  SDL_FreeSurface(scroll_surface);
  SDL_FreeSurface(img_surface);

  SDL_Quit( );

  exit( code );
}
```

Dit is de essentiële afsluit functie, welke alle bronnen vrijgeeft die we gebruikten en `SDL_Quit` aanroept.

```
void handle_key_down( SDL_keysym* keysym )
{
    switch( keysym->sym )
      {
      case SDLK_ESCAPE:
        quit( 0 );
        break;
      default:
        break;
      }
}
```

Een "Toets ingedrukt" gebeurtenis: in dit geval de ESC toets.

```
void process_events( void )
{
    SDL_Event event;

    while( SDL_PollEvent( &event ) ) {

        switch( event.type ) {
        case SDL_KEYDOWN:
            handle_key_down( &event.key.keysym );
            break;
        case SDL_QUIT:
            quit( 0 );
            break;
        }
    }
}
```

De niet minder essentiële gebeurtenis management functie.

```
void init()
{
  SDL_Surface* tmp;
  Uint16 i;

  tmp = SDL_CreateRGBSurface(SDL_HWSURFACE, SCREEN_WIDTH,
       SCREEN_HEIGHT, 8, r_mask, g_mask, b_mask, a_mask);

  scroll_surface = SDL_DisplayFormat(tmp);


  SDL_FreeSurface(tmp);
```

Laten we eens werken met een `tmp` oppervlak om `scroll_surface` te initiëren en `temp_surface` . Beide zijn geconverteerd in het video framebuffer formaat door de `SDL_DisplayFormat` functie.

```
  img_surface = IMG_Load("img.pcx");
```

Hier laden de image in het bestand in img_surface.

```
  for (i = 0; i < SDL_NUMEVENTS; ++i)
    {
      if (i != SDL_KEYDOWN && i != SDL_QUIT)
        {
          SDL_EventState(i, SDL_IGNORE);
        }
    }

  SDL_ShowCursor(SDL_DISABLE);
}
```

Alle gebeurtenissen worden genegeerd, behalve een toetsaanslag en een exit van het programma; verder schakelen we de cursor uit.

```
int main( int argc, char* argv[] )
{
```

```
  SDL_Rect ScrollFrom = {1, 0, SCREEN_WIDTH, SCREEN_HEIGHT};
  SDL_Rect First = {0, 0, 1, SCREEN_HEIGHT};
  SDL_Rect Last = {SCREEN_WIDTH - 1, 0, 1, SCREEN_HEIGHT};
```

Dit zijn de drie rechthoeken genoemd in het artikel.

```
  if (!set_video(SCREEN_WIDTH, SCREEN_HEIGHT, 8,
      SDL_HWSURFACE | SDL_HWACCEL | SDL_HWPALETTE /*| SDL_FULLSCREEN*/))
   quit(1);

  SDL_WM_SetCaption("Demo", "");

  init();

  SDL_BlitSurface(img_surface, 0, scroll_surface, 0);
```

Deze code initialiseert het hele gebeuren: zet de video mode, schrijft de venster titel, roept `init()` aan en bereid `scroll_surface` voor.

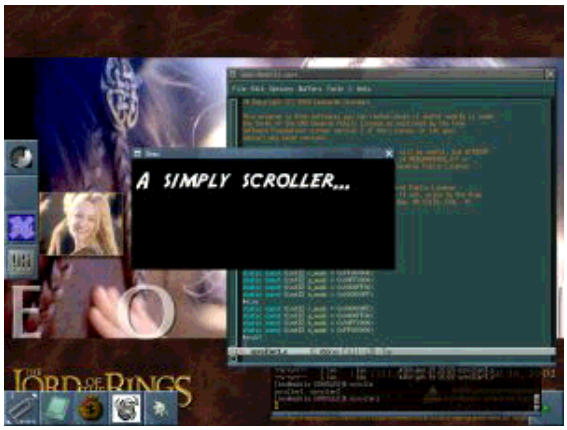
```
  while( 1 )
    {
      process_events();

      SDL_BlitSurface(scroll_surface, &First, temp_surface, 0);

      SDL_BlitSurface(scroll_surface, &ScrollFrom, scroll_surface, 0);

      SDL_BlitSurface(temp_surface, &First, scroll_surface, &Last);

      SDL_BlitSurface(scroll_surface, 0, screen, 0);

      SDL_Flip(screen);
    }

  return 0;
}
```

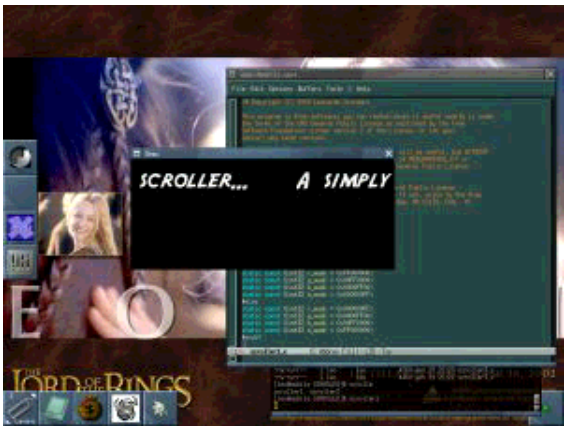
Dit is de loop die in het artikel werd beschreven: alleen de gebeurtenis controle functies en de scherm oppervlak flip zijn toegevoegd.

Zoals je kunt zien, is het initialisatie werk voor de library niet kort, maar het voordeel is dat het standaard is voor de hele demo, dus terwijl de code toeneemt, de initialisatie zal zelfs meer een klein deel van het volledige programma zijn en is opnieuw te gebruiken en te porten.

Een demo

Hier is een screenshot van de scroller op verschillende tijden:





ps: Commentaar, correcties en vragen kun je sturen naar mijn mail adres of via de Talkback pagina. Schrijf me aub in Engels, Duits of Italiaans.

Site onderhouden door het LinuxFocus editors team

http://www.LinuxFocus.org

Vertaling info:
it --> -- : Leonardo Giordani <leo.giordani(at)libero.it>
it --> en: Leonardo Giordani <leo.giordani(at)libero.it>
en --> nl: Guus Snijders <ghs(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51