# Corso di Grafica Computazionale

## Libreria SDL

**Docente:**
**Massimiliano Corsini**

*Laurea Specialistica in Ing. Informatica*

*Facoltà di Ingegneria*
*Università degli Studi di Siena*

# Libreria SDL



- La ***libreria SDL*** *(Simple DirectMedia Library)* nasce per sviluppare facilmente videogiochi amatoriali
- Multipiattaforma: Linux, MacOS, BeOS, Win32
- Low-level API (ma non troppo...)
- Composta da otto moduli:
  - Video
  - Event Handling
  - Multi-Threading
  - Timers
  - Audio
  - File I/O
  - CDROM
  - Joystick

# Applicazioni Event-Based



- Struttura classica programmi a linea di comando:

```
main()
{
  init();
  // Algoritmo...
  exit();
}
```

- Struttura programmi event-based:

```
main()
{
  init();
  while (true) {
    get_event();
    process_event();
  }
  exit();
}
```

# Inizializzare SDL



- Dobbiamo inizializzare SDL indicando i moduli che si vogliono utilizzare. Ad esempio se la nostra applicazione utilizzerà il modulo video ed i timer dovremo scrivere:

```
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_TIMER);
```

- Quando non vogliamo più utilizzare SDL possiamo disattivare la gestione di tutti i sottosistemi (audio, video, ...) con

```
SDL_Quit();
```

# Esempio Applicazione SDL



```
#include "SDL.h"      /* SDL.h va incluso in ogni
                         applicazione SDL */
#include <stdio.h>
int main()
{
   /* Initializzazione SDL + Video + Audio */
   if ((SDL_Init(SDL_INIT_VIDEO|SDL_INIT_AUDIO)==-1))
   {
      printf("Could not initialize SDL: %s.\n",
     SDL_GetError());  /* stampa dell'errore */
      exit(-1);
   }
   printf("SDL initialized.\n");

   /* Shutdown all subsystems */
   SDL_Quit();

   exit(0);
}
```

# SDL Video



- E’ il modulo più completo (anche perchè è quello più utilizzato).

```
/*
 * Initialize the display in a 640x480 8-bit
 * palettized mode,requesting a software surface
 */
 SDL_Surface *screen;
 screen = SDL_SetVideoMode(640, 480, 8,
SDL_SWSURFACE);
 if ( screen == NULL )
 {
  fprintf(stderr, "Couldn’t set 640x480x8 video
          mode: %s\n", SDL_GetError());
   exit(1);
 }
```

# SDL Video



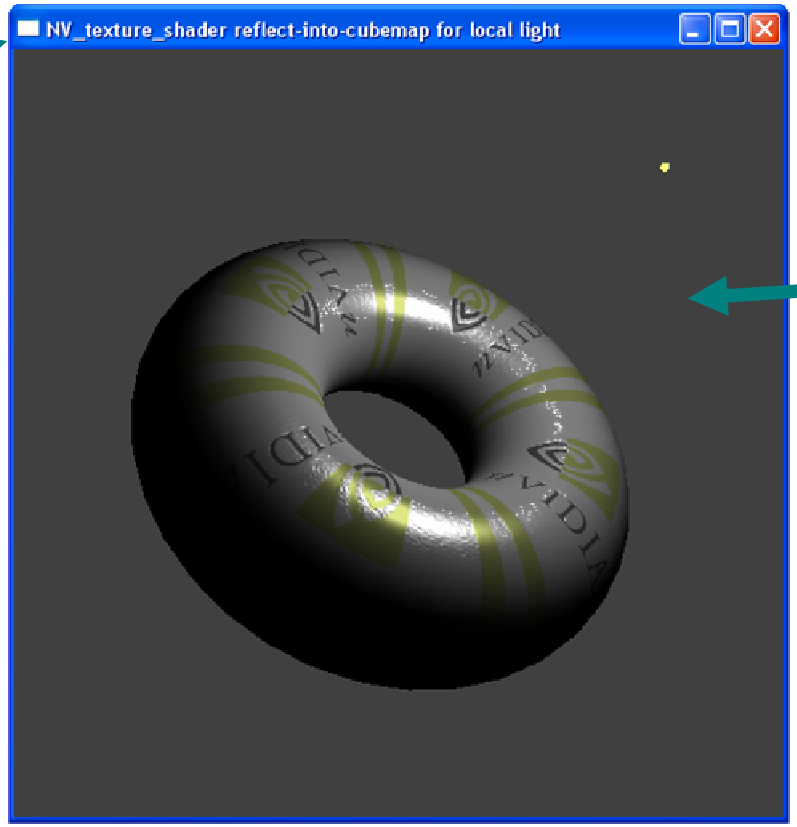
- Sulle SDL_Surface è possibile scrivere e leggere pixel (per esempio copiare un'immagine per visualizzarla).
- L'SDL supporta il caricamento ed il salvataggio di immagini bitmap.
  - ```
    SDL_Surface *image;
    /* Load the BMP file into a surface */
    image = SDL_LoadBMP("image.bmp");
    ```
- Per salvare:
  - ```
    SDL_SaveBMP(SDL_Surface *surface, const
    char *filename);
    ```

# SDL & OpenGL



- SDL è in grado di creare contesti di rendering OpenGL su differenti piattaforme (Linux/X11, Win32, MacOs X, Solaris/X11, ...).

Finestra di Sistema




OpenGL Rendering Context

# SDL & OpenGL



- Inizializzare un contesto di rendering OpenGL non è molto diverso da inizializzare una modalità video.
- **`SDL_SetVideoMode(width, height, bpp, SDL_OPENGL | SDL_FULLSCREEN);`**
- Prima di inizializzare OpenGL si devono csettare gli attributi che si desiderano.
- Se lavoriamo in finestra anzichè a tutto schermo dobbiamo stare attenti a settare i bit-per-pixel (bpp) a seconda del modo video corrente.

# SDL & OpenGL



- Ad esempio, settiamo la profondità di colore; vogliamo almeno 5 bit per la componente RED, 5 per la componente GREEN e 5 quella BLUE:

```
SDL_GL_SetAttribute( SDL_GL_RED_SIZE, 5 );
SDL_GL_SetAttribute( SDL_GL_GREEN_SIZE, 5 );
SDL_GL_SetAttribute( SDL_GL_BLUE_SIZE, 5 );
SDL_GL_SetAttribute( SDL_GL_DEPTH_SIZE, 16 );
```

- Abilitiamo il DOUBLE BUFFER:

```
SDL_GL_SetAttribute( SDL_GL_DOUBLEBUFFER, 1 );
```

# Input Handling



- Cominciamo con il vedere le strutture dati per gestire *l'input da tastiera*:
  - ***SDLKey*** è un enum definito in SDL/include/SDL_keysym.h che definisce i vari simboli. Ad esempio **SDLK_a** corrisponde al tasto 'a', **SDLK_SPACE** corrisponde alla barra spaziatrice e così via.
  - ***SDLMod*** enumera i modificatori (Ctrl, Shift, Alt).
  - ***SDL_keysym*** è una struttura dati contenente tutte le informazioni relative allo stato della tastiera.

# Input Handling



- ***SDL_KeyboardEvent*** è una struttura dati contenente il tipo dell'evento (tasto premuto o tasto rilasciato) e lo stato attuale della tastiera (*SDL_keysym*).

# Keyboard Events



```
SDL_Event event;

/* Poll for events. SDL_PollEvent() returns 0 when there are
 * no more events on the event queue, our while loop wil
 * exit when that occurs.
 */
while( SDL_PollEvent( &event ) )
{
  switch( event.type )
  {
    case SDL_KEYDOWN:
      printf( "Key press detected\n" );
    break;
    case SDL_KEYUP:
      printf( "Key release detected\n" );
    break;
    default:
    break;
  }
}
```

# Keyboard Events



- Per poter leggere quali caratteri sono stati premuti è opportuno settare la conversione dall'UNICODE (**`SDL_EnableUNICODE(1)`**).
- SDL_GetKeyName(...) ritorna il nome in forma leggibile del carattere premuto
- In realtà a noi interessa il simbolo:

  **`event.key.keysym.sym ==`** *sdlsymbol*

  *(esempio: SDLK_LEFT, SDLK_RIGHT)*

# Mouse Events



```
SDL_Event event;
while ( SDL_PollEvent(&event) )
{
  switch (event.type)
  {
    case SDL_MOUSEMOTION:
      printf("Il Mouse e' stato mosso da (%d,%d) a
              (%d,%d)\n",
              event.motion.xrel, event.motion.yrel,
              event.motion.x, event.motion.y);
    break;
    case SDL_MOUSEBUTTONDOWN:
      printf("Il pulsante %d del Mouse %d e' stato
         premuto a (%d,%d)\n", event.button.button,
         event.button.x, event.button.y);
  break;
  case SDL_QUIT:
    exit(0);
  }
}
```

# Eventi SDL



- La struttura dati SDL_Event contiene informazioni sul ***tipo*** e *tutte* le strutture dati relative agli eventi.

- ```
  typedef union
  {
    Uint8 type;
    SDL_ActiveEvent active;
    SDL_KeyboardEvent key;
    SDL_MouseMotionEvent motion;
    SDL_MouseButtonEvent button;
    ...
    SDL_JoyButtonEvent jbutton;
    SDL_ResizeEvent resize;
    SDL_QuitEvent quit;
  } SDL_Event;
  ```
  ```

# Eventi SDL



- Quindi, come abbiamo visto, la gestione degli eventi è semplice:
  - si estrae l'evento dalla coda degli eventi (SDL_PollEvent)
  - si fa uno switch sul tipo dell'evento e si gestisce opportunamente l'evento in base al tipo.
- Sottolineamo che SDL prevede anche delle funzioni ad hoc per gestire la coda degli eventi:
  - **`SDL_PollEvent(...)`**
  - **`SDL_WaitEvent(...)`**
  - **`SDL_SetEventFIlter(...)`**
  - **`SDL_PeepEvents(...)`**
  - Ecc..

# Domande?