

GLUT: GL Utility Toolkit

E' una API utile per realizzare programmi *window independent* per il *rendering* con OpenGL .

Funzionalità:

- finestre multiple per *rendering*;
- gestione degli eventi;
- supporto della funzione *idle*;
- gestione di menu a cascata;
- gestione generale delle finestre di visualizzazione;

<http://www.xmission.com/~nate/glut.html>

<http://www.opengl.org/developers/documentation/glut.html>

GLUT (2)

Terminologia

callback: puntatore alla routine di gestione, chiamata al verificarsi di un determinato evento. (fase di registration);

idle: è uno stato in cui il window system non riceve eventi di alcun tipo (il sistema non fa nulla);

pop-up menu: menù che appare alla pressione di un tasto del mouse;

Top-level, sub window: una top-level window è ridimensionabile, si può spostare..., una subwindow può essere creata solo all'interno di una top-level window.

GLUT (3)

Fase di INIZIALIZZAZIONE

prefisso: **glutInit__** vengono chiamate una sola volta all'interno del programma.

void glutInit(int *argc, char ** argv): inizializza la libreria e attiva una sessione con il window system della macchina.

void glutInitWindowSize(int width, int height)

void glutInitWindowPosition(int x, int y): specificano la dimensione della finestra e la posizione iniziale espresse in pixel.

GLUT (4)

void glutInitDisplayMode(unsigned int mode):
inizializza le caratteristiche di visualizzazione.

mode: maschera di bit messi in OR.

Modalità principali:

GLUT_RGB: colori espressi in RGB;

GLUT_INDEX: colori espressi in modo indicizzato;

GLUT_SINGLE: modalità di buffer singolo;

GLUT_DOUBLE: modalità di buffer doppio;

GLUT_DEPTH: buffer di profondità, per z-buffering.

...

GLUT (5)-ciclo eventi

`void glutMainLoop(void)`: attiva il ciclo che catturerà gli eventi.

Viene chiamata solo una volta e ha lo scopo di gestire tutti quegli eventi per i quali esiste una **callback** function che sia stata registrata.

GLUT (6)-esempio

Esempio:

```
void InitGLUT(void){
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition( 50, 50 );
    glutInitWindowSize( 300, 300 );
    ...
    ...
}

void main(int argc, char *argv[]){
    glutInit(&argc, argv);
        InitGLUT(); //init GLUT environment
        InitGL(); //init OpenGL environment
    glutMainLoop();
    exit(0);
}
```

GLUT (7)-gestione finestre

Glut gestisce finestre di *top-level* e *sub-window*:

int glutCreateWindow(char *name): crea una finestra di top-level intitolata "name". Restituisce un identificatore unico.

int glutCreateSubWindow(int win, int x, int y, int width, int height): crea una sub-window all'interno di una top-level window identificata da *win*.

A ogni finestra (top-level oppure sub-window) è associato un contesto OpenGL che deve essere inizializzato immediatamente dopo la creazione di una finestra.

L'ultima finestra creata è impostata a *current-window*.

GLUT (8)-gestione finestre

`void glutSetWindow(int win)`

`int glutGetWindow(void):`

rispettivamente permettono di impostare win come *current-window*, e di reperire la *current-window* in uso.

`void glutDestroyWindow(int win):`

distrugge la finestra indentificata da win e il suo contesto OpenGL.

`void glutPostRedisplay(void):`

genera un evento che permette di ridisegnare il contenuto della *current-window*

`glutPositionWindow(int x, int y):` riposizionano la *current-window*;

`glutReshapeWindow(int width, int height):` ridimensionano la c.w.;

`glutFullScreen(void):` la c.w. ha le dimensioni dello schermo.

GLUT(9)-esempio

```
#include<GL/glut.h>

int m_w; //main window;

void InitGLUT(void){
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowPosition( 50, 50 );
glutInitWindowSize( 300, 300 );
m_w = glutCreateWindow( "My first window" );
}

void main(int argc, char *argv[]){
    glutInit(&argc, argv);
        InitGLUT(); //init GLUT environment
        InitGL(); //init OpenGL environment
    glutMainLoop();
exit(0);
}
```

GLUT (10)

`void glutSetCursor(int cursor)`: imposta l'immagine del cursore del mouse quando questo si trova sulla curret-window.

Valori di *cursor* possono essere:

GLUT_CURSOR_RIGHT_ARROW;

GLUT_CURSOR_LEFT_ARROW;

GLUT_CURSOR_INFO;

GLUT_CURSOR_HELP;

GLUT_CURSOR_WAIT;

...

GLUT (11) pop-up menu

La libreria GLUT permette di creare menù a comparsa nel momento in cui il sistema riceve come evento il click di un bottone del mouse.

int glutCreateMenu(void (*func)(int value)):

func: callback function che restituisce un void e ha come parametro un int (voce selezionata del menu). Restituisce un indice al menù corrente: *current-menu*.

glutAddMenuEntry(char *name, int value): aggiunge una voce di menu al current-menu;

void glutAttachMenu(int button)

void glutDetachMenu(int button): associa/toglie la comparsa del menu al bottone del mouse indentificato da *button* (GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, GLUT_RIGHT_BUTTON).

Esempio Menu

```
#define WIREFRAME      100
#define SHADED        101
#define COLOR          102
#define LIGHT         103
int id_MENU1;
void menu1(int id_entry)
{
    switch(id_entry){
        case(WIREFRAME):
            printf("ID_entry: %d\n",id_entry);
            break;
        case(SHADED):
            printf("ID_entry: %d\n",id_entry);
            break;
        .....
        .....
        default:
            break;
    }
}
```

Esempio Menu (2)

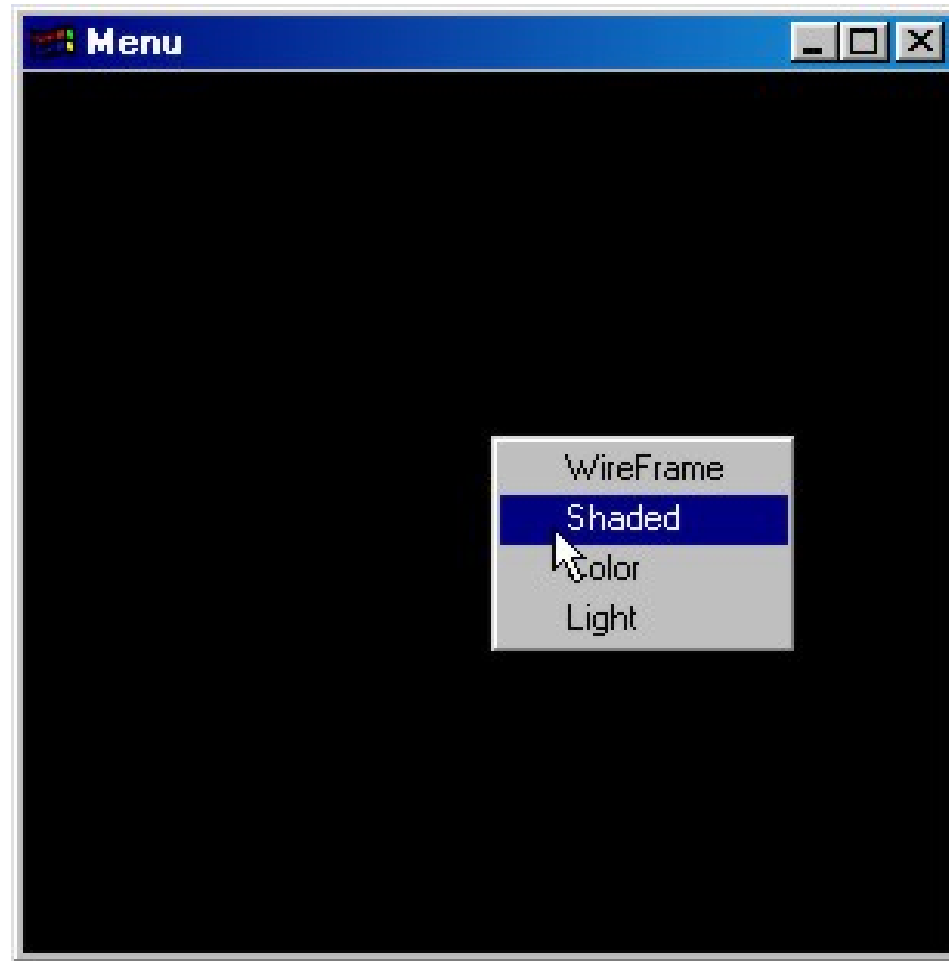
```
void InitGlut(void)
{
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutInitWindowSize(300,300);
    glutInitWindowPosition(50,50);
    glutCreateWindow("Menu"); //Creo TOP-LEVEL Window
// Creazione MENU
// Creo il CURRENT-MENU
    id_MENU1=glutCreateMenu(menu1);
// Inserisco le Entry
    glutAddMenuEntry("WireFrame", WIREFRAME);
    glutAddMenuEntry("Shaded", SHADED);
    glutAddMenuEntry("Color", COLOR);
    glutAddMenuEntry("Light", LIGHT);

    glutAttachMenu(GLUT_RIGHT_BUTTON);

    /*Registrazione CALL-BACK function*/
    glutDisplayFunc(mainWin);
}
```

Esempio Menu (3)

Risultato:



GLUT (12) sub-menu

E' possibile aggiungere un sotto menu al *current-menu* principale:

void glutAddSubMenu(char *name, int menu):

name: voce che appare nel menu;

menu: id del menu da aggiungere al *current-menu*;

Esempio sub-menu

```
.....
```

```
id_MENU1=glutCreateMenu(menu1); //Creo il CURRENT-MENU  
glutAddMenuEntry("Wire Frame",WIREFRAME); //Inserisco le Entry  
glutAddMenuEntry("Color",COLOR);  
glutAddMenuEntry("Light",LIGHT);
```

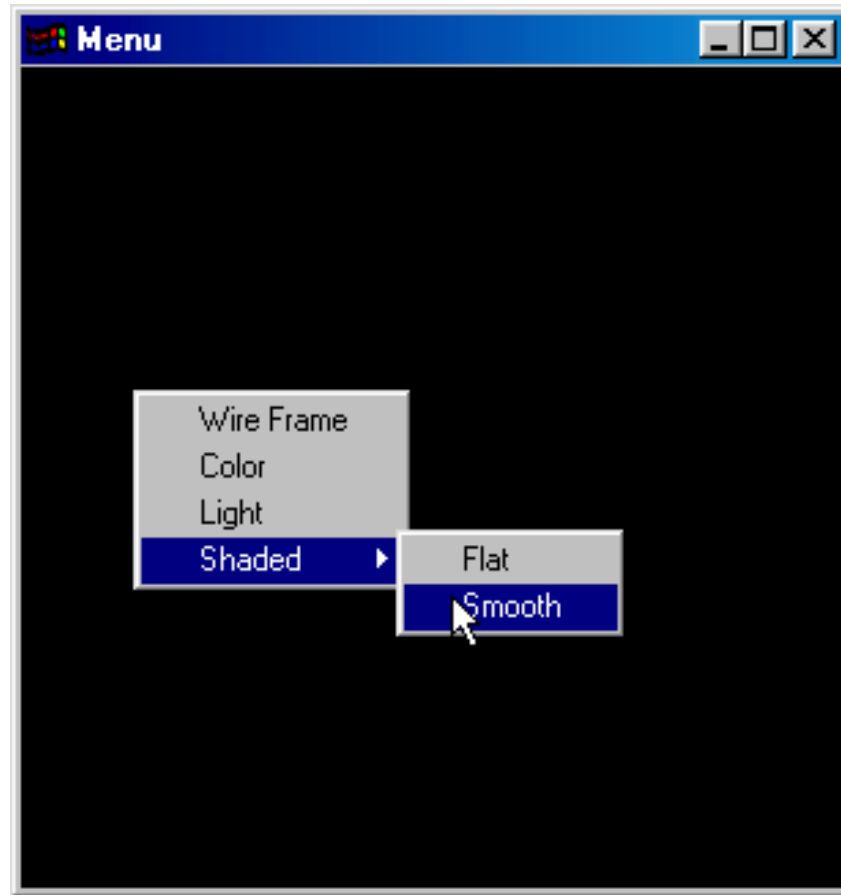
```
id_MENU2=glutCreateMenu(menu1); //Creo Sub-MENU  
glutAddMenuEntry("Flat",SHADED_FLAT); //Aggiungo Voci Sub-Menu  
glutAddMenuEntry("Smooth", SHADED_SMOOTH);
```

```
glutSetMenu(id_MENU1); //Imposto il CURRENT-MENU  
glutAddSubMenu("Shaded",id_MENU2); //Aggiungo il Sub-MENU  
glutAttachMenu(GLUT_RIGHT_BUTTON);
```

```
.....
```

Esempio sub-menu (2)

Risultato:



GLUT fase di registrazione

glutMainLoop() attiva un processo server che riceve i messaggi generati in fase di esecuzione (mouse, tastiera resize, redisplay, ...).

A ciascuno di essi è associato una funzione di gestione (handler, callback-function).

```
int [void] main(int argc, char *argv[]){  
    glutInit(&argc, argv);  
    InitGLUT();  
    InitGLUI();  
    InitGL();  
    glutMainLoop();  
}
```

Inizializzazione/Creazione
Registrazione

GLUT (13) registrazione

Funzioni di callback registration hanno la seguente
Struttura: `glutTYPEFunc(handler *func(param));`

TYPE: tipo di evento;

func: puntatore alla funzione che deve essere eseguita
per gestire l'evento.

● GLUT (14) registrazione

`void glutDisplayFunc(void (*func)(void))`: è la più importante. Gestisce la visualizzazione della c.w. Ogni finestra deve avere una callback function registrata per l'evento di Display. Tale registrazione va eseguita prima che la finestra venga visualizzata.

```
#include <GL/glut.h>
int win1;
void displaywin1(void){
    glClear(BUFFER);
    glClearColor(R,G,B);
    glFlush();
}
int main(::::){
    win1=glutCreateWindow("NAME");
    glutDisplayFunc(displaywin1);
}
```

● GLUT (15) registrazione

void glutReshapeFunc(void(*func)(int width, int height)):
associata al ridimensionamento della current-window.

```
void m_wReshape(int w, int h){
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if(w<=h)
        glOrtho(-1.0,1.0, -1.0*(GLfloat)h/(GLfloat)w, 1.0*(GLfloat)h/(GLfloat)w,-1.0,1.0);
    else
        glOrtho(-1.0*(GLfloat)w/(GLfloat)h, 1.0*(GLfloat)w/(GLfloat)h,-1.0,1.0,-1.0,1.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Utilizzata per il ridimensionamento della viewport di visualizzazione.

GLUT (16) registrazione

Ordine DISPLAY - RESHAPE

glutMainLoop();

ricezione msg:
inizializzazione,
creazione:
glutCreateWindow()

glutReshape()

primo display()

GLUT (17) registrazione

Eventi del mouse

void glutMouseFunc(void(*func)(int b, int state, int x,inty)):
cattura l'evento associato alla pressione di un bottone del mouse, all'interno della c.w..

b: GLUT_LEFT_BUTTON, GLUT_RIGHT_BUTTON ...
indica quale bottone e' stato premuto.

state: GLUT_DOWN, GLUT_UP;

x,y: posizione del mouse all'interno della c.w.

GLUT (17) eventi mouse

`void glutMotionFunc(void(*func)(int x, int y))`

`void glutPassiveMotionFunc(void(*func)(int x, int y)):`

Cattura la posizione del mouse mentre lo si muove all'interno di una window. La differenza sta nel tenere premuto o meno un bottone.

`void glutEntryFunc(void(*func)(int state)):` associato all'evento di entrata/uscita del mouse all'interno della c.w.;
state: GLUT_LEFT, GLUT_ENTERED.

GLUT (18) Tastiera

void glutKeyboardFunc(void(*func)(unsigned char k, int x, int y)):

cattura l'evento associato alla pressione di un tasto della tastiera.

key: indica quale tasto è stato premuto.

x, y: indicano la posizione del mouse all'interno della c.w.

void glutSpecialFunc(void(*func)(unsigned char k, int x, int y)):

cattura la pressione dei tasti speciali: F1, F2 ...F12, tasti direzionali, pageup, pagedown, end, insert, ...

GLUT (18)

Concludendo la API di GLUT permette di interfacciarsi al window system di una macchina e di gestire gli eventi base associati all'hardware di base della macchina e quelli associati alle modifiche della GUI vera e propria.

Ciascuna *callback function* è legata alla current-window in uso, di conseguenza se ho più finestre, è necessario prima attivarle come c.w. (`glutSetWindow()`) e quindi registrare le relative *callback functions*.