Knowledge
Aided
Engineering
Manufacturing
and
Related
Technologies

Dipartimento di
Ingegneria
Industriale

Università
di Parma

introduzione a

OpenGL

esercitazione 1

INTRODUZIONE

- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING

INTRODUZIONE

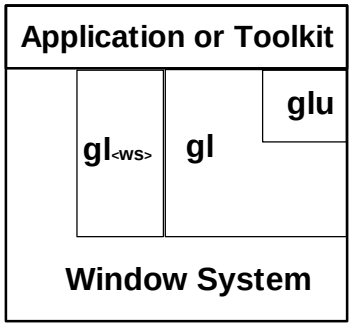
- OpenGL
  - e` una libreria grafica 3D
  - e` uno standard grafico
  - e` platform-independent
  - e` window system independent
  - e` hardware-independent
  - e` in grado di gestire HW grafico sofisticato
  - ne esistono implementazioni per X, NT, OS/2, ...
  - permette di scrivere codice portabile

INTRODUZ. (cont.)

- Essendo indipendente da HW e SW, OpenGL non tratta:
  - la gestione di eventi
  - l'accesso alla tastiera
  - l'accesso al mouse
  - l'accesso ad altri dispositivi
  - il window management
- Inoltre:
  - non e` di alto livello
  - non e` Object-Oriented
  - si programma in C language
  - e` una API
  - e` una macchina a stati

API e LIBRERIE

- L'ambiente di sviluppo per OpenGL e` normalmente costituito da alcune librerie:
  - **`gl`** OpenGL
  - **`glu`** Utility
  - **`gl<ws>`** Estensione per **`<ws>`**
  - **`glut`** Interfaccia a **`<ws>`**

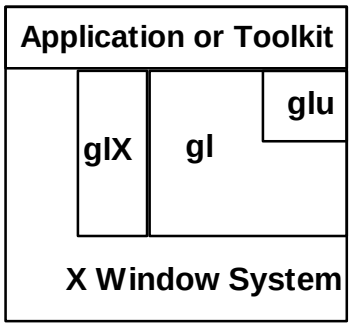


LIBRERIE: gl, glu

- **`gl`**
  - rendering 3D
  - lighting
  - z-buffering
  - alpha blending
  - texture mapping
  - antialiasing
  - fog
- **`glu`**
  - gestione parametri viewing
  - gestione texture mapping
  - polygon tessellation (decompositore generico di poligoni concavi)
  - curve e superfici parametriche
  - gestione errori

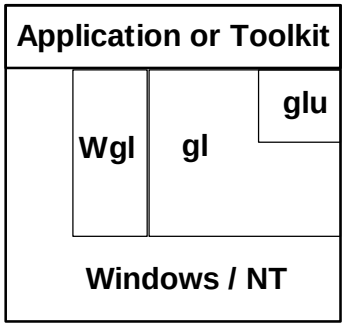
LIBRERIE: glX

- **`glX`**
  - "estensione formale" di X
  - accesso ai font di X
  - gestione delle pixmaps



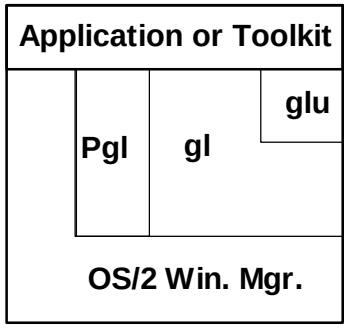
LIBRERIE: Wgl

- **`Wgl`**
  - gestione metafiles
  - accesso ai font di Windows
  - supporto printing



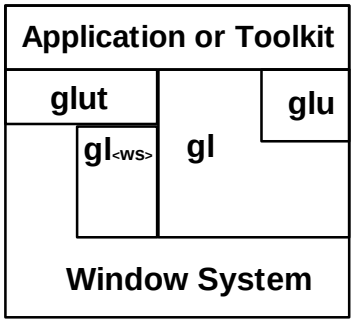
LIBRERIE: Pgl

- **`Pgl`**
  - estensione per OS/2
  - ...
  - ...



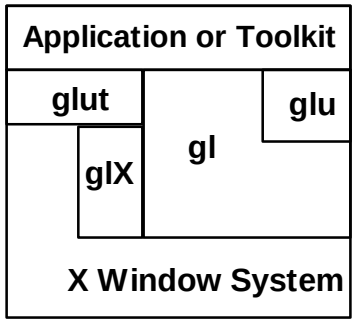
LIBRERIE: glut

- **`glut`**
  - interfaccia con il window system
  - gestione eventi
  - gestione input keyboard/mouse
  - primitive 3D
  - realizza trasparenza rispetto al window system sottostante

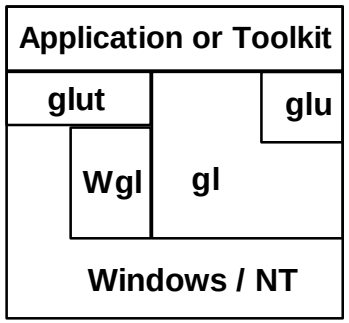


LIBRERIE: glut (cont.)

- **`glut`** e X-Windows



- **`glut`** e MS-Windows



- INTRODUZIONE
- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING

INDICE ESERCITAZIONI

- “hello world”
- initializing the window
- color index mode
- rgb color mode
- clearing the buffers
- drawing primitives
- completing operations

FORMATO COMANDI

- Il formato generale e`

  `<lib_name><Command_name><signature>`

  dove **`<signature>`** e`

  `[[<arg_number>]<arg_type>] ... ]`

- esempi:

  `glColor3f`

  libreria **`gl`**, colore dati 3 float

  `glVertex2i`

  libreria **`gl`**, vertice dati 2 int

  `glutCreateWindow`

  libreria **`glut`**, crea la window

TIPI DI DATI

tipo	abbrev.	lungh.
GLbyte	b	8 bit
GLubyte	ub	8 bit
GLshort	s	16 bit
GLushort	us	16 bit
GLint	i	32 bit
GLuint	ui	32 bit
GLfloat	f	32 bit
GLdouble	d	64 bit

OPZIONI DI COMPILAZIONE

- include directories

```
-I/usr/local/include
```

- library directories

```
-L/usr/local/lib
-L/usr/X11/lib
```

- link libraries

```
-lX11 -lXext -lXmu -lXi
-lglut -lGLU -lGL
-lm
```

HEADER FILES

- per utilizzare OpenGL

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

- per utilizzare la libreria glu

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

- per utilizzare la libreria glut

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


ESERCITAZIONE - 1

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


void redraw( void )
{
   glClear( GL_COLOR_BUFFER_BIT );

   glBegin( GL_POINTS );
   glVertex2f( 0.0, 0.0 );
   glEnd();

   glFlush();
}


void main( int argc, char** argv )
{
   glutCreateWindow( "Window Title" );

   glutDisplayFunc( redraw );
   glutMainLoop();
}
```

INIZIALIZZ. WINDOW

```
void glutInitDisplayMode( unsigned int mask )
```

- definisce la modalita` della window
- valori disponibili per definire **`mask`**
  - `GLUT_RGB`
  - `GLUT_INDEX`
  - `GLUT_SINGLE`
  - `GLUT_DOUBLE`
  - `GLUT_DEPTH`
  - `GLUT_STENCIL`
  - `GLUT_ACCUM`
  - `GLUT_STEREO`
- esempio di utilizzo:

```
glutInitDisplayMode( GLUT_RGB | GLUT_DEPTH )
```


INIZ. WINDOW (cont.)

```
void glutInitWindowPosition(int x, int y)
```

- definisce la posizione iniziale della window

```
void glutInitWindowSize(int width, int height)
```

- definisce le dimensioni iniziali della window

```
void glutCreateWindow( char *title )
```

- crea la window secondo modalita` e geometria specificate

ESERCITAZIONE - 2

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

void redraw( void )
{
   glClear( GL_COLOR_BUFFER_BIT );

   glBegin( GL_POINTS );
   glVertex2f( 0.0, 0.0 );
   glEnd();

   glFlush();
}

void main( int argc, char** argv )
{
   glutInitDisplayMode   ( GLUT_RGB );
   glutInitWindowPosition( 300, 100 );
   glutInitWindowSize    ( 150, 150 );
   glutCreateWindow      ( argv[0]  );

   glutDisplayFunc( redraw );
   glutMainLoop   ();
}
```


IL COLORE

- OpenGL possiede due modalita`:
  - modalita` indicizzata
  - modalita` rgb
- utilizzo modalita` indicizzata:
  - selezionare la modalita`
    `glutInitDisplayMode( GLUT_INDEX );`
  - definire le entry della LUT
    `glutSetColor( index, r, g, b );`
  - settare il colore per il buffer clearing
    `glClearIndex( index );`
  - settare il colore corrente
    `glIndexi( index );`
  - il colore corrente e` uno stato

ESERCITAZIONE - 3

```
#include <GL/glut.h>
void redraw( void )
{
   glutSetColor( 0, 0.0, 0.4, 0.0 );
   glutSetColor( 1, 1.0, 0.0, 1.0 );

   glPointSize( 3.0 );
   glClearIndex( 0 );
   glClear( GL_COLOR_BUFFER_BIT  );
   glBegin( GL_POINTS );
      glIndexi( 1 );
      glVertex2f( 0.0, 0.0 );
   glEnd();
   glFlush();
}

void main( int argc, char** argv )
{
   glutInitDisplayMode   ( GLUT_INDEX );
   glutInitWindowPosition( 300, 100   );
   glutInitWindowSize    ( 150, 150   );
   glutCreateWindow      ( argv[0]    );

   glutDisplayFunc( redraw );
   glutMainLoop   (        );
}
```


IL COLORE (cont.)

- utilizzo modalita` rgb:
  - selezionare la modalita`
    `glutInitDisplayMode( GLUT_RGB );`
  - settare il colore per il buffer clearing
    `glClearColor( r, g, b, 1.0 );`
  - settare il colore corrente
    `glColor3f( r, g, b );`
  - il colore corrente e` uno stato

ESERCITAZIONE - 4

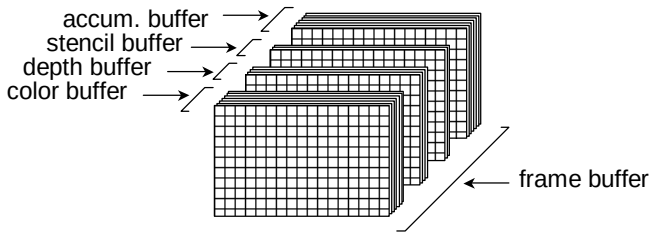
```
#include <GL/glut.h>
void redraw( void )
{
   glPointSize( 3.0 );
   glClearColor( 0.0, 0.4, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );
   glBegin( GL_POINTS );
      glColor3f( 1.0, 0.0, 1.0 );
      glVertex2f( 0.0, 0.0 );
   glEnd();
   glFlush();
}

void main( int argc, char** argv )
{
   glutInitDisplayMode   ( GLUT_RGB );
   glutInitWindowPosition( 300, 100 );
   glutInitWindowSize    ( 150, 150 );
   glutCreateWindow      ( argv[0]  );

   glutDisplayFunc ( redraw );
   glutMainLoop    (        );
}
```


IL FRAME BUFFER

- Il frame buffer di OpenGL e` costituito da piu`buffers:
  - color buffer
  - depth buffer
  - stencil buffer
  - accumulation buffer



CLEARING BUFFERS

- Prima di cominciare a disegnare e` necessario inizializzare i buffer che si intendono utilizzare

- Per ogni buffer e` possibile specificare il valore con cui esso viene inizializzato, ad es. per specificare il colore iniziale del color buffer:

- ```
  glClearColor( r, g, b, a );
  ```

  oppure

  ```
  glClearIndex( i );
  ```

  a seconda della modalita` utilizzata.

CLEAR. BUFFERS (cont.)

- la funzione che inizializza i buffer selezionati al valore stabilito e`

  ```
  glClear( GLbitfield mask )
  ```

  dove **`mask`** puo` valere

  ```
  GL_COLOR_BUFFER_BIT
  GL_DEPTH_BUFFER_BIT
  GL_STENCIL_BUFFER_BIT
  GL_ACCUMULATION_BUFFER_BIT
  ```

- esempio:

```
glClearColor( 0.0,0.0,1.0,1.0 );
glClear(  GL_COLOR_BUFFER_BIT |
          GL_DEPTH_BUFFER_BIT );
```

inizializza il color buffer a blu,
e lo z-buffer al valore di default.
```

PRIMITIVES

- le primitive di OpenGL (punti, linee, triangoli, poligoni) sono descritte in termini di vertici
- i vertici sono descritti in coordinate omogenee

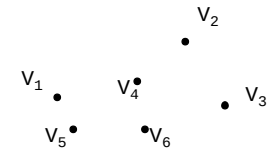
  (x, y, z, w) = (x/w, y/w, z/w, 1)

```
glVertex4f( x, y, z, w ) (x, y, z,  w  )
glVertex3f( x, y, z )    (x, y, z,  1.0)
glVertex2f( x, y )       (x, y, 0.0,1.0)
```

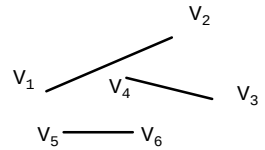
- il formato generale per il drawing e`:

```
glBegin( tipo_di_primitiva );
  <sequenza di vertici>
glEnd();
```

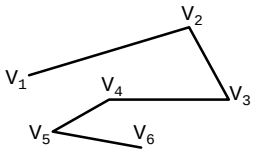

PRIMITIVE TYPES



GL_POINTS



GL_LINES



GL_LINE_STRIP

PRIMITIVE TYPES (cont.)

- esempio:

```
glBegin( GL_LINES );
   glColor3f( 1.0, 1.0, 1.0 );
   glVertex2f( -0.8, -0.8 );
   glColor3f( 1.0, 0.0, 0.0 );
   glVertex2f(  0.5,  0.2 );
   glVertex2f( -0.5,  0.4 );
   glColor3f( 0.0, 1.0, 0.0 );
   glVertex2f( -0.2,  0.1 );
glEnd();
```

- vertici differenti possono avere colore differente; in tal caso il colore viene interpolato da un vertice all'altro

ESERCITAZIONE - 5

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

void redraw( void )
{
   glPointSize( 3.0 );

   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_POINTS );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex3f(  0.5,  0.2, -0.3 );
      glVertex4f( -0.5,  0.4,  0.7, 1.0 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex4f( -0.5,  0.4,  0.7, 3.0 );
   glEnd();

   glFlush();
}

...
```

ESERCITAZIONE - 6

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

void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_LINES );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex3f(  0.5,  0.2, -0.3 );
      glVertex4f( -0.5,  0.4,  0.7, 1.0 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex4f( -0.5,  0.4,  0.7, 3.0 );
   glEnd();

   glFlush();
}

...
```


ESERCITAZIONE - 7

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

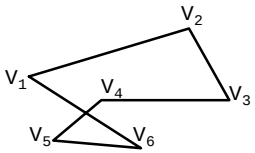
void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_LINE_STRIP );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex3f(  0.5,  0.2, -0.3 );
      glVertex4f( -0.5,  0.4,  0.7, 1.0 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex4f( -0.5,  0.4,  0.7, 3.0 );
   glEnd();

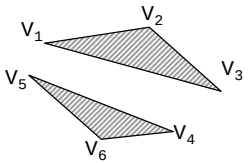
   glFlush();
}

...
```

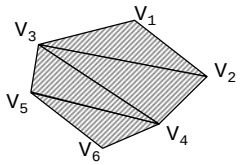
PRIMITIVE TYPES (cont.)



GL_LINE_LOOP



GL_TRIANGLES



GL_TRIANGLE_STRIP

ESERCITAZIONE - 8

```
...

void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_LINE_LOOP );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex3f(  0.5,  0.2, -0.3 );
      glVertex4f( -0.5,  0.4,  0.7, 1.0 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex4f( -0.5,  0.4,  0.7, 3.0 );
   glEnd();

   glFlush();
}

...
```

ESERCITAZIONE - 9

```
...

void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_TRIANGLES );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex2f(  0.5,  0.2 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex2f( -0.16,  0.11 );

      glColor3f( 0.0, 0.0, 1.0 );
      glVertex2f( -0.5,  0.6 );
      glVertex2f( -0.2,  0.3 );
      glVertex2f(  0.6,  0.8 );
   glEnd();

   glFlush();
}

...
```


ESERCITAZIONE - 10

```
...

void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

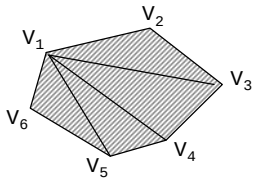
   glBegin( GL_TRIANGLE_STRIP );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( -0.8, -0.8 );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex2f(  0.5,  0.2 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex2f( -0.16,  0.11 );

      glColor3f( 0.0, 0.0, 1.0 );
      glVertex2f( -0.5,  0.6 );
      glVertex2f( -0.4,  0.0 );
   glEnd();

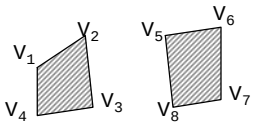
   glFlush();
}

...
```

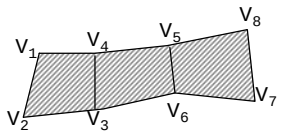
PRIMITIVE TYPES (cont.)



GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP

ESERCITAZIONE - 11

```
...

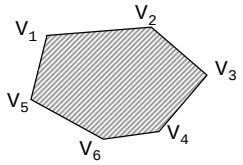
void redraw( void )
{
  glClearColor( 0.0, 0.0, 0.0, 1.0 );
  glClear( GL_COLOR_BUFFER_BIT  );

  glBegin( GL_TRIANGLE_FAN );
     glColor3f( 1.0, 1.0, 1.0 );
     glVertex2f( 0.0, 0.0 );
     glColor3f( 1.0, 0.0, 0.0 );
     glVertex2f(  0.8,  0.2 );
     glColor3f( 0.0, 1.0, 0.0 );
     glVertex2f( 0.7,  0.5 );

     glColor3f( 0.0, 0.0, 1.0 );
     glVertex2f( -0.5,  0.6 );
     glVertex2f( -0.7,  0.0 );
     glColor3f( 1.0, 1.0, 0.0 );
     glVertex2f( -0.5, -0.6 );
  glEnd();

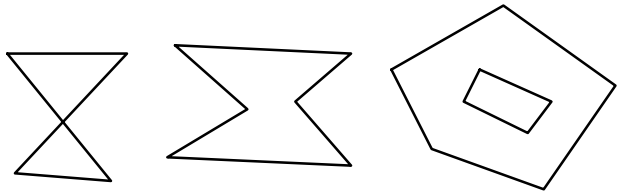
  glFlush();
}
...
```

PRIMITIVE TYPES (cont.)



GL_POLYGON

- requisiti di OpenGL sui poligoni:
  - convessi
  - vertici coplanari
  - perimetro non autointersecantesi



Poligoni non validi !!

ESERCITAZIONE - 12

```
...

void redraw( void )
{
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glBegin( GL_POLYGON );
      glColor3f( 1.0, 0.0, 0.0 );
      glVertex2f(  0.8,  0.2 );
      glColor3f( 0.0, 1.0, 0.0 );
      glVertex2f( 0.7,  0.5 );
      glColor3f( 0.0, 0.0, 1.0 );
      glVertex2f( -0.5,  0.6 );
      glVertex2f( -0.7,  0.0 );
      glColor3f( 1.0, 1.0, 0.0 );
      glVertex2f( -0.5, -0.6 );
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex2f( 0.5, -0.5 );
   glEnd();

   glFlush();
}

...
```

COMPLETARE LE OPERAZIONI

- **`glFlush`**`()`

  richiede che venga iniziato lo svuotamento della pipeline e cede il controllo all'istruzione successiva. Assicura che le operazioni in corso vengano completate in un tempo finito.

- **`glFinish`**`()`

  richiede che venga iniziatato lo svuotamento della pipeline e attende che sia completato prima di cedere il controllo all'itruzione successiva.

© F. DE ANGELIS