Corso di Grafica Computazionale

OpenGL

Docente:
Massimiliano Corsini

Laurea Specialistica in Informatica

Facoltà di Ingegneria
Università degli Studi di Siena

Introduzione

- Open Graphics Language
  - Libreria C
  - Cross-platform
  - Qualche centinaio di routines
  - Specifiche definite dall'***ARB***
  - Sito di riferimento: ***www.opengl.org***

Breve Storia

- Inizialmente sviluppato da Silicon Graphics

- Ora:
  OpenGL **A**rchitecture **R**eview **B**oard
  - mantiene e aggiorna le *specifiche*
  - versione attuale: **3.0**
  - una compagnia, un voto

- Ci sono anche le *estensioni* private
  - Soprattutto ATi e nVIDIA.

GLU e GLUT

- OpenGL è il layer di base
- GLU (GL Utilites)
  - insieme di funzioni di utility costruite sopra OpenGL, piu'comode da usare
  - esempio
```
void gluLookAt(eyex,eyey,eyez,
               cx,cy,cz,
               upx, upy, upz);
```

- GLUT e' il Toolkit di interfaccia con il SO
- Wgl e GLx sono i sottoinsiemi di OpenGL che dipendono dal SO

Sintassi

- Tutte le funzioni di OpenGL hanno la sintassi tipo:
  - glFunctionXXX
- XXX specifica il tipo dei parametri
  - Esempio:
    ```
    glColor3f(float, float, float);
    glColor3fv(float*);
    ```
    f: float d: double v: vettore

    b: byte i: integer ecc…
  - Non e' C++ …

Macchina a Stati

- L'OpenGL funziona come una ***macchina a stati***:
  - colore corrente
  - posizione luci
  - Matrici di trasformazione

  fanno parte dello stato corrente
- Molti comandi OpenGL modificano soltanto lo stato corrente

Matrici in OpenGL

Sono memorizzate per colonne => ***column-major order***

$$\begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$$

Matrici e stato

- Lo stato comprende due matrici (e due stacks)
  - Matrice *Model-View*
  - Matrice di *Proiezione*
- Una di queste è sempre la matrice di lavoro (matrice corrente sulla quale avvengono le operazioni)
- Per cambiare la matrice di lavoro si utilizza
  - `glMatrixMode(GL_MODELVIEW);`
  - `glMatrixMode(GL_PROJECTION);`

Matrici e stato

- Per rimpiazzare la matrice di lavoro
  - `glLoadIdentity();`
  - `glLoadMatrixf(float *m);`
- Tutti i comandi che operano sulla matrice di lavoro la modificano moltiplicandola per un’altra matrice

Trasformazioni geometriche

- Rotazioni
  - `glRotatef(angle, ax, ay, az);`
- Traslazioni
  - `glTranslatef(tx,ty,tz);`
- Scalature (non uniformi)
  - `glScalef(sx,sy,sz);`
- Generica
  - `glMultMatrixf(float f*);`

Operazioni su Matrici

- Vista:
  - `void gluLookAt(eyex,eyey,eyez, cx,cy,cz, upx, upy, upz);`
- Proiezione:
  - `glOrtho2D(left, right, bottom, top);`
  - `void gluPerspective(fovy, aspect, zNear, zFar);`

Stack di Matrici

- Operazioni sullo stack
  - `glPushMatrix();`
  - `glPopMatrix();`
- Esempio di utilizzo:
  - glMatrixMode(GL_PROJECTION);
  - Push M_{prsp}
  - Carico una matrice di proiezione ortogonale per disegnare uno sfondo
  - Disegno lo sfondo
  - Pop M_{prsp}
  - Disegno la scena sullo sfondo

Inviare triangoli

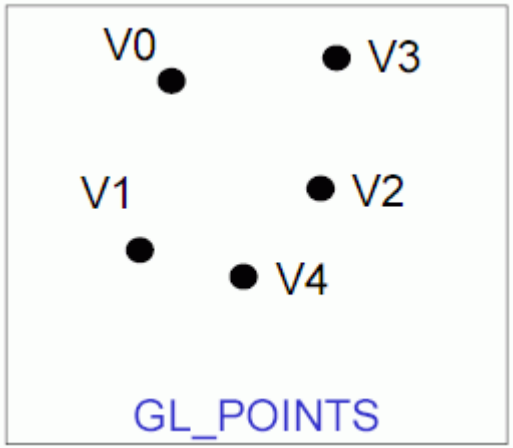

```
glBegin(GL_TRIANGLES);

glVertex3d(x1,y1,z1);
glVertex3d(x2,y2,z2);
glVertex3d(x3,y3,z3);

glVertex3d(x4,y4,z4);
glVertex3d(x5,y5,z5);
glVertex3d(x6,y6,z6);

glVertex3d(x7,y7,z7);
glVertex3d(x8,y8,z8);
glVertex3d(x9,y9,z9);
...
glEnd();
```

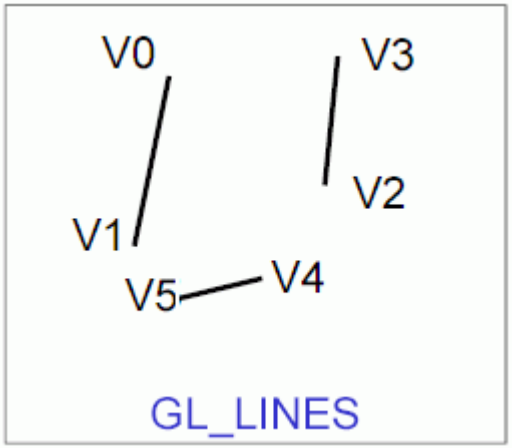
primo triangolo

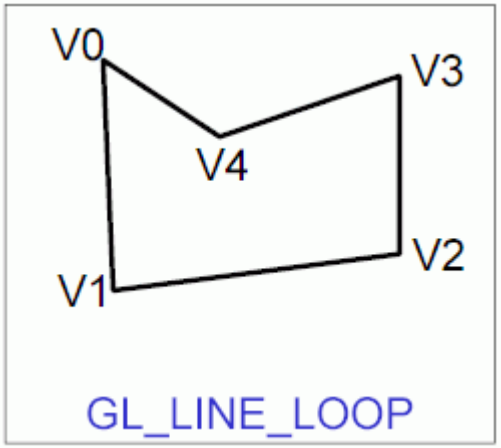
secondo triangolo

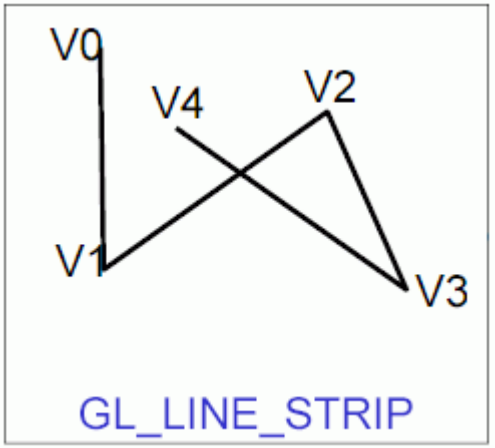
terzo triangolo

Inviare punti e linee

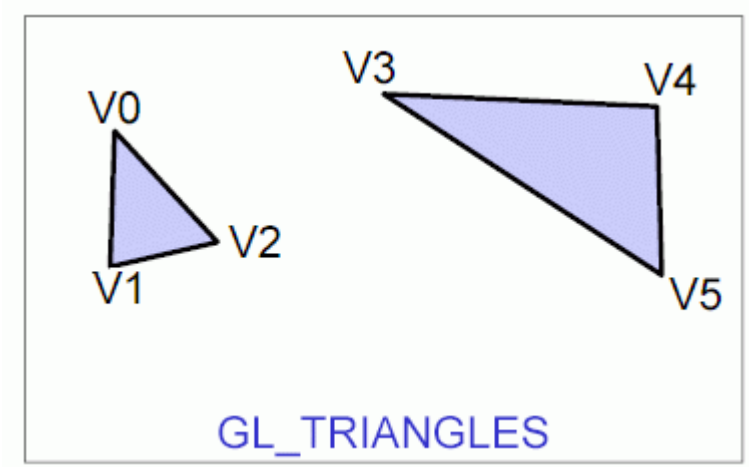


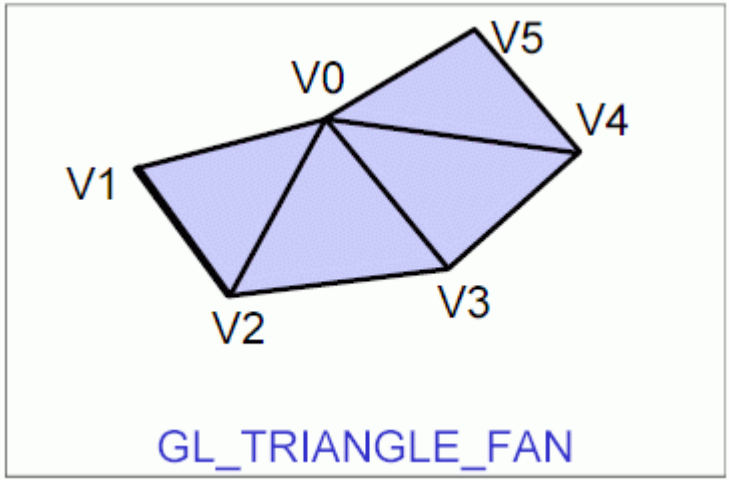


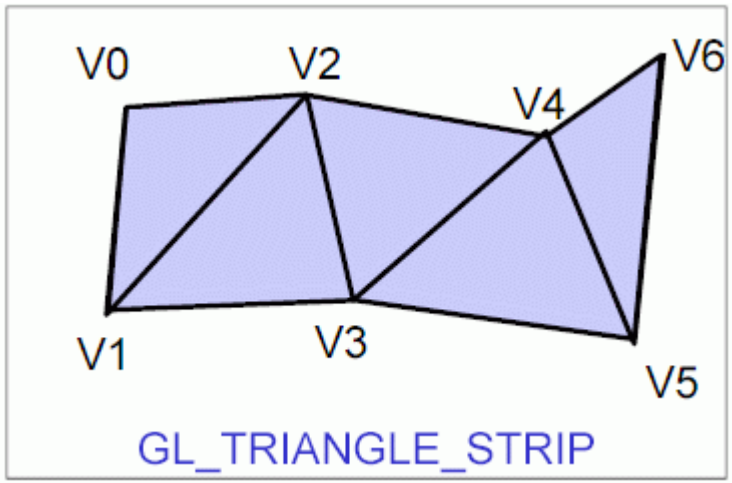




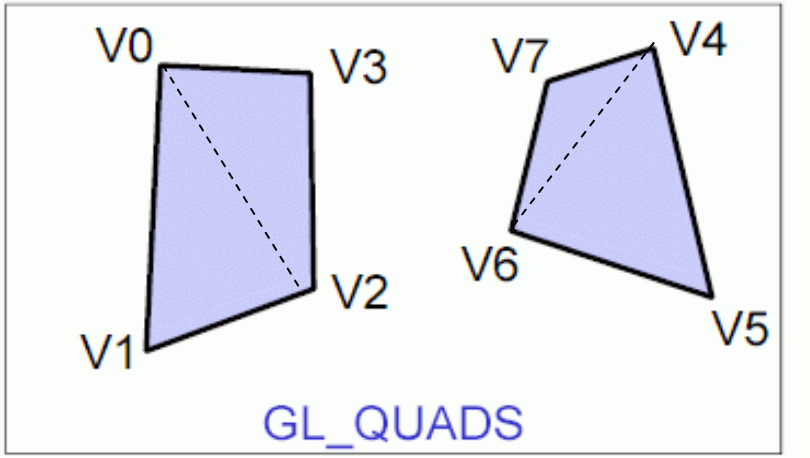
Inviare Fan e Strip

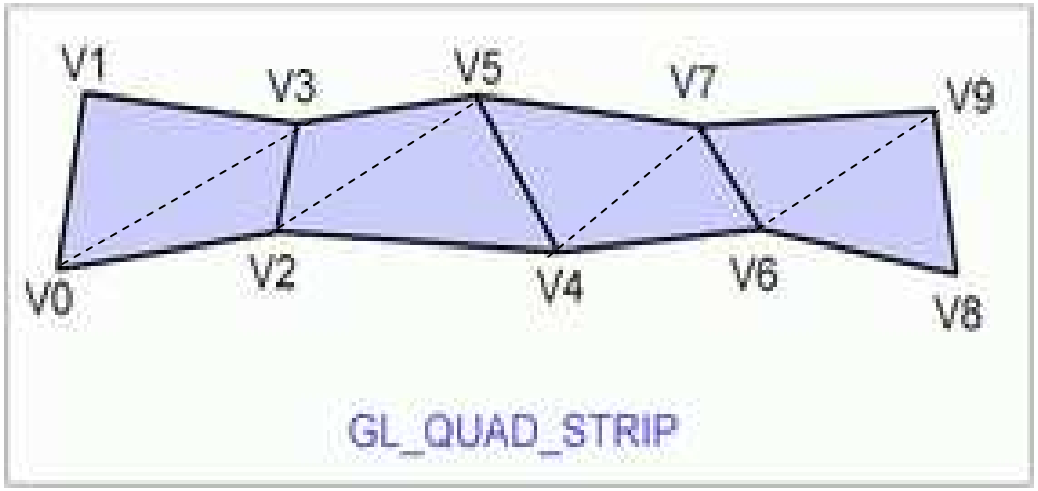






Inviare Quads





Face Culling

- ***`void glFrontFace(GLenum mode);`***
- Determina come una faccia viene considerata front-face oppure back-face. La modalità ***GL_CCW***, considera come front-face quelle i cui vertici proiettati sono ordinati in senso antiorario. ***GL_CW*** in senso orario.
- ***`void glCullFace(GLenum mode);`***
- Permette di settare su quali tipi di facce deve essere eseguito il *culling*.
- Le possibili modalità sono ***GL_FRONT***, ***GL_BACK*** e ***GL_FRONT_AND_BACK***.
- Il comando **`glEnable(GL_CULL_FACE)`** abilita il culling. **`glDisable(GL_CULL_FACE)`** lo disabilita.

Materiali ed Illuminazione

- **Generalità sulla gestione del colore**
  - Come già detto lavoriamo nello spazio RGB
  - Se una luce ha colore (LR, LG, LB), ed il materiale ha colore (MR, MG, MB), il colore risultante (il colore della luce che arriva all'osservatore) sarà ottenuto moltiplicando i due colori: (LR*MR, LG*MG, LB*MB).
  - Se due fonti luminose risultano nei colori *osservati* (già moltiplicati per il materiale) (R1, G1, B1) ed (R2, G2, B2), l'OpenGL sommerà tali contributi ottenendo (R1+R2, G1+G2, B1+B2).
  - Valori maggiori di 1 vengono troncati ad 1. Infatti il valore uno corrisponde idealmente al massimo dell'intensità che il dispositivo di visualizzazione può riprodurre per quel dato colore.

Materiali ed Illuminazione

1. Calcolo delle normali (per facce o per vertice).
   → Noi assumiamo le normali ai vertici già definite e calcolate correttamente.
2. Creazione e posizionamento delle sorgenti di luce.
3. Definizione delle proprietà del materiale e dei parametri del modello di illuminazione.

Creazione e posizionamento sorgenti di luce

- ***void glLight{if}(GLenum light, GLenum pname, TYPE param);***

  void glLight{if}v(GLenum light, GLenum pname, TYPE *param);
- Crea la sorgente di luce specificata dal tag light: ***GL_LIGHT0***, ***GL_LIGHT1***, ... , ***GL_LIGHT7***.
- Si possono definire al massimo 8 luci.
- ***pname*** specifica il parametro da settare.
- La versione che usa i vettori deve essere utilizzata quando i dati relativi alla luce (ad esempio il colore) sono memorizzati appunto in un vettore.

Creazione e posizionamento sorgenti di luce (esempi)

- Settaggio della posizione della luce:

```
GLfloat light_position[] =
        {1.0, 1.0, 1.0, 0.0};
glLightfv(GL_LIGHT0, GL_POSITION,
 light_position);
```

- Settaggio della componente di ambiente della:

```
GLfloat light_ambient[] =
        {0.0, 0.0, 0.0, 1.0};
glLightfv(GL_LIGHT0, GL_AMBIENT,
 light_ambient);
```

Creazione e posizionamento della luce (esempi)

- Settaggio della componente diffusiva:

```
GLfloat light_diffuse[] =
        {1.0, 1.0, 1.0, 1.0};
glLightfv(GL_LIGHT0, GL_DIFFUSE,
 light_diffuse);
```

- Settaggio della componente speculare:

```
GLfloat light_specular[] =
        {1.0, 1.0, 1.0, 1.0};
glLightfv(GL_LIGHT0, GL_SPECULAR,
 light_specular);
```

Luci puntuali e direzionali

- Esistono due tipi di luci: ***posizionali*** e ***direzionali***.
- La posizione della luce è data in coordinate omogenee. Quindi è possibile posizionare luci all'infinito → ***luci direzionali***.
- Le luci direzionali sono utili per modellare il sole o comunque sorgenti luminose che, per la loro elevata distanza, incidono uniformemente sulla superficie del modello.

Attenuazione della luce

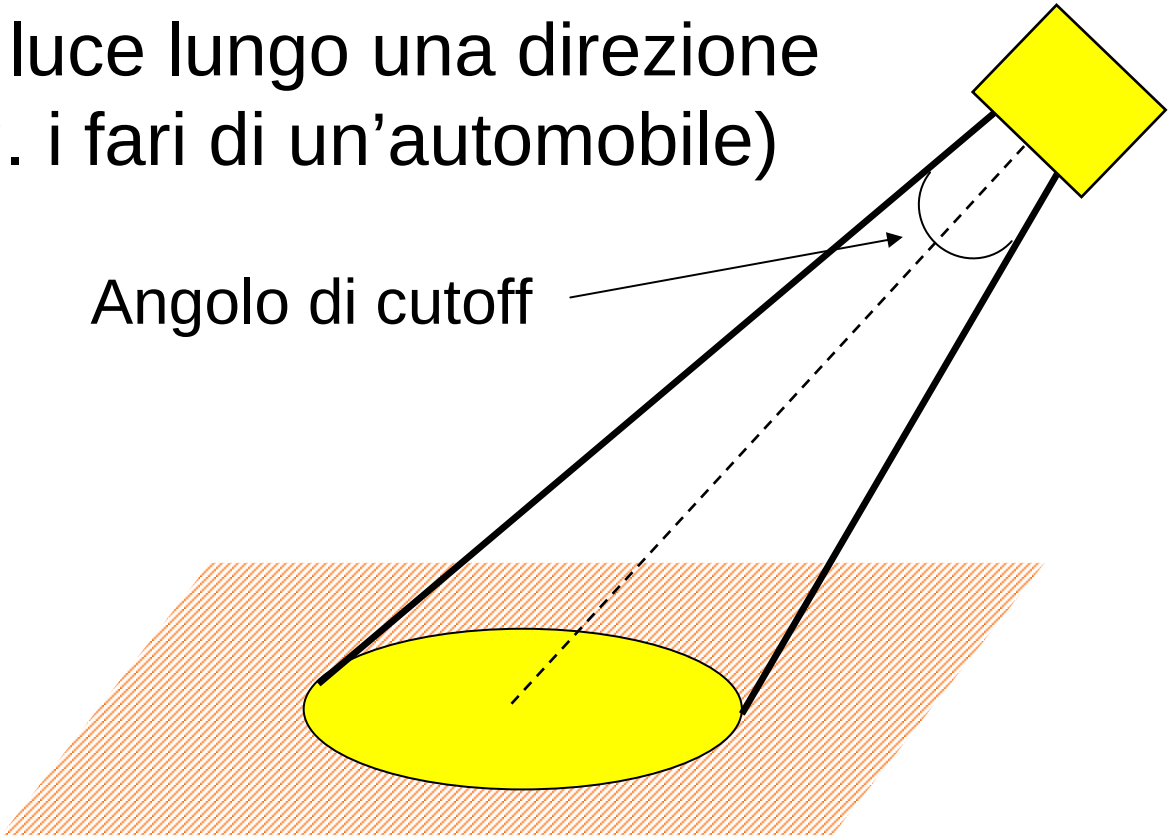
- Settaggi dell'attenuazione della luce:

```
glLightf(GL_LIGHT0,
   GL_CONSTANT_ATTENUATION, 2.0);
glLightf(GL_LIGHT0,
   GL_LINEAR_ATTENUATION, 1.0);
glLightf(GL_LIGHT0,
   GL_QUADRATIC_ATTENUATION, 0.5);
```

- Per una luce direzionale non ha senso...

Spotlight

- Le sorgenti luminose di tipo spotlight modellano sorgenti luminose che concentrano la luce lungo una direzione privilegiata (e.g. i fari di un'automobile)



Spotlight ed OpenGL

- E' possibile settare la posizione dello spotlight (**GL_POSITION**), l'angolo di cut-off (**GL_SPOT_CUTOFF**), la direzione dello spot (**GL_SPOT_DIRECTION**) e l'attenuazione della luce dal centro al bordo dello spot (**GL_SPOT_EXPONENT**).
- La formula che regola l'illuminazione all'interno del cono di luce è la seguente:

$$I = I_c(\vec{P} \cdot \vec{D})\mathrm{GL\_EXPONENT}$$

Proprietà dei Materiali

- Si utilizza principalmente il comando **`glMaterialfv(...);`**
- Ad esempio, volendo settare la risposta del materiale alle componenti di ambiente, diffusiva e speculare si può scrivere:

```
GLfloat mat_ambient[] = { 0.7, 0.7, 0.7, 1.0 };
GLfloat mat_diffuse[] = { 0.1, 0.5, 0.8, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv(GL_FRONT, GL_SPECULAR,
mat_specular);
```

- In pratica questi sono i famosi ***Ka***, ***Kd*** e ***Ks*** che avevamo visto nel modello di illuminazione di Phong.

Proprietà dei Materiali

- Ricordiamoci che la specularità del materiale è caratterizzata anche dall'***esponente di riflessione speculare (n)***
- Per specificarlo:

```
GLfloat mat_shininess[] = {50.0f};
glMaterialfv(GL_FRONT,
     GL_SHININESS, mat_shininess);
```

Abilitazione del lighting

- Si utilizza il solito **`glEnable(..)`** / **`glDisable(..)`**
- Si deve anche speficare quali luci devono essere abilitate
- Esempio:

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHT1);
...
glDisable(GL_LIGHT0);glDisable(GL_LIGHT1);
glDisable(GL_LIGHTING);
```

Texture Mapping

1. Creare una texture (texture object) e specificarne i parametri.
2. Indicare la modalità di applicazione della texture.
3. Abilitare il Texture Mapping
4. Disegnare la scena inviando oltre alle primitive geometriche le coordinate textures.

Creazione Texture

A. Generazione del *nome* (handle) della texture.
B. Bind (creazione) del texture object con i dati, incluse le proprietà.
C. Settare le priorità nel caso l'hardware supporti un working set di textures ad elevate performances.
D. Durante l'uso dell'applicazione si effettuano operazioni di ***bind*** e ***rebind*** degli oggetti texture, per rendere tali dati disponibili durante la fase di visualizzazione della scena.

Nomi della Texture

- ***void glGenTextures(GLsizei n, GLuint *textureNames);***
- Restituisce n nomi di texture inutilizzati. I codici restituiti possono non essere contigui.
- Il valore zero è riservato e non può mai essere restituito come *texture name*.
- Esempio:

```
GLUint texHandle = 0;
glGenTextures(1, &texHandle);
```

Creazione ed uso Texture

- ***void glBindTexture(GLenum target, GLuint textureName);***
- Quando *textureName* è utilizzato per la prima volta, un nuovo oggetto texture è creato e collegato a tal nome.
- Quando il binding riguarda una texture precedentemente creata essa diventa attiva.
- Quando il valore di textureName è zero, OpenGL interrompe l'uso delle textures. fault texture.

Esempio Creazione Texture e Settaggio Parametri


```
GLuint texName = 0;
glGenTextures(2, &texName);
glBindTexture(GL_TEXTURE_2D, texName);

// SETTAGGIO PARAMETRI DI WRAPPING
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
   GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
   GL_CLAMP);

// SETTAGGIO PARAMETRI DI FILTRAGGIO
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
   GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
   GL_NEAREST);

// SETTAGGIO DEI DATI
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkImageWidth,
   checkImageHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE,
   checkImage);
```

Abilitare Texture Mapping

- Si utilizza il solito **`glEnable(..)`** / **`glDisable(..)`**

- Esempio:

```
glEnable(GL_TEXTURE_2D);
...uso il texture mapping...
glDisable(GL_TEXTURE_2D);
```

Inviare Coordinate Texture


```
glBindTexture(GL_TEXTURE_2D, &texName);
glBegin(GL_TRIANGLES);

// primo triangolo
glTexCoord2f(u1,v1);
glVertex3f(x1,y1,z1);
glTexCoord2f(u2,v2);
glVertex3f(x2,y2,z2);
glTexCoord2f(u3,v3);
glVertex3f(x3,y3,z3);

// secondo triangolo
glTexCoord2f(u4,v4);
glVertex3f(x4,y4,z4);
glTexCoord2f(u5,v5);
glVertex3f(x5,y5,z5);
glTexCoord2f(u6,v6);
glVertex3f(x6,y6,z6);

glEnd();
```

Modalità di applicazione della Texture

- ***void glTexEnv{if}(GLenum target, GLenum pname, TYPE param);***
- ***void glTexEnv{if}v(GLenum target, GLenum pname, TYPE *param);***
- Setta la modalità di applicazione della texture.
- Il parametro ***target*** deve essere GL_TEXTURE_ENV.
- Se ***pname*** vale GL_TEXTURE_ENV_MODE, ***param*** può assumere i valori GL_DECAL, GL_REPLACE, GL_MODULATE, o GL_BLEND, andando a specificare come i valori di colore delle texture devono essere combinati con i valori di colore dei frammenti processati.
- Se ***pname*** vale GL_TEXTURE_ENV_COLOR, ***param*** deve essere un array composto da quattro floating-point rappresentati le componenti R, G, B, and A. Questa modalità è utilizzata in caso GL_BLEND sia stata precedentemente specificata.

Modalità di applicazione delle Texture

- L’effetto dipende anche dal formato interno della texture (GL_RGB, GL_RGBA, ecc).
- Tendenzialmente GL_DECAL sostituisce il colore della texture.
- GL_REPLACE compie operazioni simili.
- GL_MODULATE usa il colore della texture per modulare quello del frammento.
- Nello spefico le funzioni potete trovarle sul RedBook.

Modalità di applicazione delle Texture

- REPLACE:
  - GL_RGB → C = Ct , A = Af
  - GL_RGBA → C = Ct , A = At
- DECAL:
  - GL_RGB → C = Ct , A = Af
  - GL_RGBA → C = Cf (1 – At) + Ct At , A = Af
- MODULATE
  - GL_RGB → C = CfCt , A = Af
  - GL_RGBA → C = CfCt , A = AfAt

Blending

- È una tecnica di imaging che ci permette di mescolare i colori di più immagini
- Per la precisioni i colori del framebuffer vengono mescolati con i colori della nuova immagine che si sta generando
- Utilizzo immediato → gestione della trasparenza

Semitrasparenza

vecchia: (già sul framebuffer) | **nuova: che sovrascrivo** | **risultato**

$(1-\alpha)$ x  + α x  =  $\alpha = 0.25$

$(1-\alpha)$ x  + α x  =  $\alpha = 0.5$

$(1-\alpha)$ x  + α x  =  $\alpha = 0.75$

Alpha Blending

- I colori hanno 4 componenti:
  - R,G,B, α
- Quando arriva un frammento (sopravvissuto al depth test) invece di sovrascriverlo lo miscelo con la formula:

$$(r,g,b)_{finale} = (r,g,b)_{vecchio} \cdot (1-\alpha) + (r,g,b)_{nuovo} \cdot (\alpha)$$

NOTA: Alpha blending e Z-buffer bisticciano... l'ordine di rendering è determinante per il risultato finale.

Alpha Blending & OpenGL

- Per attivare l'alpha blending si utilizza il comando **`glEnable(GL_BLEND);`**
- Per settare la funzione di blending si utilizza il comando **`glBlendFunc(source_factor, dest_factor);`**
  - Esempio:

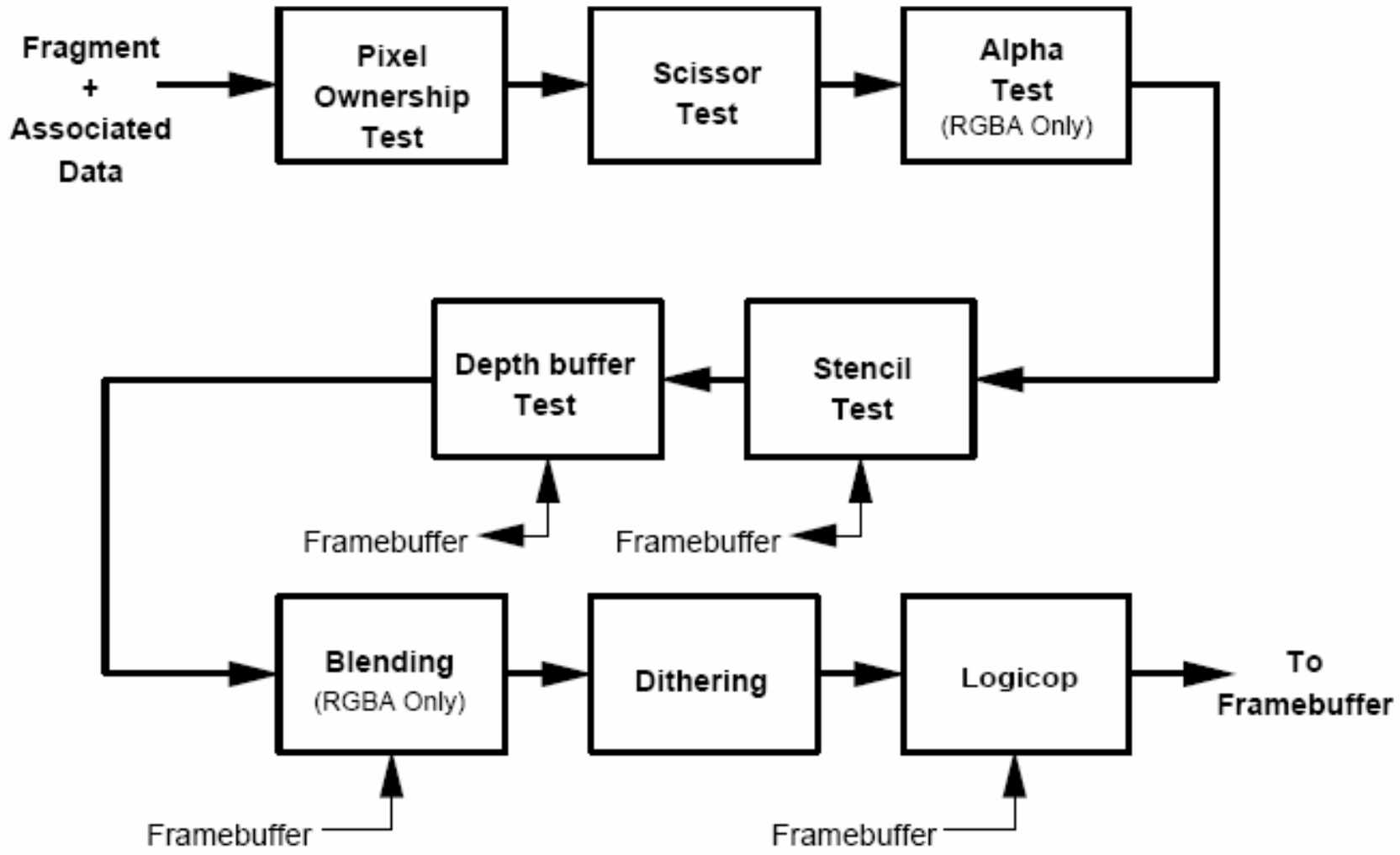
```
glBlendFunc(GL_SRC_ALPHA,
  GL_ONE_MINUS_SRC_ALPHA);
```

- Per *source* si intende l'alpha del frammento, per *dest* l'alpha di destinazione, ossia quello del framebuffer
- Ci sono molte combinazioni possibili per le funzioni di blending:
  - **GL_ZERO, GL_ONE, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_COLOR, GL_DST_ALPHA, GL_ONE_MINUS_DST_ALPHA, GL_DST_COLOR, GL_ONE_MINUS_DST_COLOR**

Per-fragment Operations

- Una volta uscito in output, il frammento subisce delle operazioni dette ***per-fragment operations*** prima di arrivare al framebuffer
- Alcune di queste operazioni sono
  - Scissor Test
  - Alpha Test
  - Stencil Test
  - Depth Test
  - Blending
  - Dithering

Per-fragment Operations



Alpha Test

- Se un frammento é troppo trasparente, scartiamolo:
  - `if ( frammento.alpha < k) then scarta frammento`
- Al solito, può essere abilitato o disabilitato con i soliti comandi OpenGL glEnable(...), glDisable(...)
  - `glEnable(ALPHA_TEST);`
- Anche la funzione di test può essere settata:
  - `glAlphaFunc(GL_GREATER, 0.01);`

Approfondimenti

- ***Redbook*** → manuale di riferimento fondamentale
- ***Nehe's Tutorial*** → piccoli programmi di esempio su ogni aspetto della pipeline
- ***www.opengl.org*** → sito ufficiale di riferimento OpenGL

Domande?