Grafica Computazionale

Lighting & Texturing in SoftOgl

Fabio Ganovelli
fabio.ganovelli@isti.cnr.it
a.a. 2005-2006

Lighting

Da Softogl.h

```
/// costanti che indicano la componente del lighting
enum Component {
AMBIENT,DIFFUSE,SPECULAR,SHININESS,EMISSION};
/// componente ambiente
extern Color4 current_ambient_color;
/// componente diffusa
extern Color4 current_diffuse_color;
/// componente speculare
extern Color4 current_specular_color;
/// esponente componente speculare
extern unsigned char  current_shininess;
/// componente emissiva
extern Color4 current_emissive_color;

/// imposta la componente specificata da c al valore color
void SetMaterial(Component c, Color4 color);
/// imposta la shininess (c deve essere SHININESS)
void SetMaterial(Component c, unsigned char esp);
```

Lighting: la classe LightType

Da Softogl.h

```
struct LightType{
    LightType(){
                position        = Point3dH(0,0,1,1);
                diffuse_color   = Color4(1.0,1.0,1.0,1.0);
                specular_color  = Color4(1.0,1.0,1.0,1.0);
                ambient_color   = Color4(0.0,0.0,0.0,1.0);
                direction       = Vector3dH(0,0,-1,0);
                exponent        = 0;
                cut_off_angle   = 180;
            };
            /// posizione della luce.
            Point3dH position;
            /// componenti diffuso,speculare e ambiente
            Color4 diffuse_color,specular_color,ambient_color;
            /// direzione dello spotlight
            Vector3dH direction;
            /// esponente
            unsigned char exponent;
            /// angolo di cut_off
            float cut_off_angle;
    };

    /// vettore contenente tutte le luci
    extern std::vector<LightType> lights;

    /// valore booleano per decidere se eseguire o no il lighting  (default a false)
    extern bool isLightingEnabled;
```

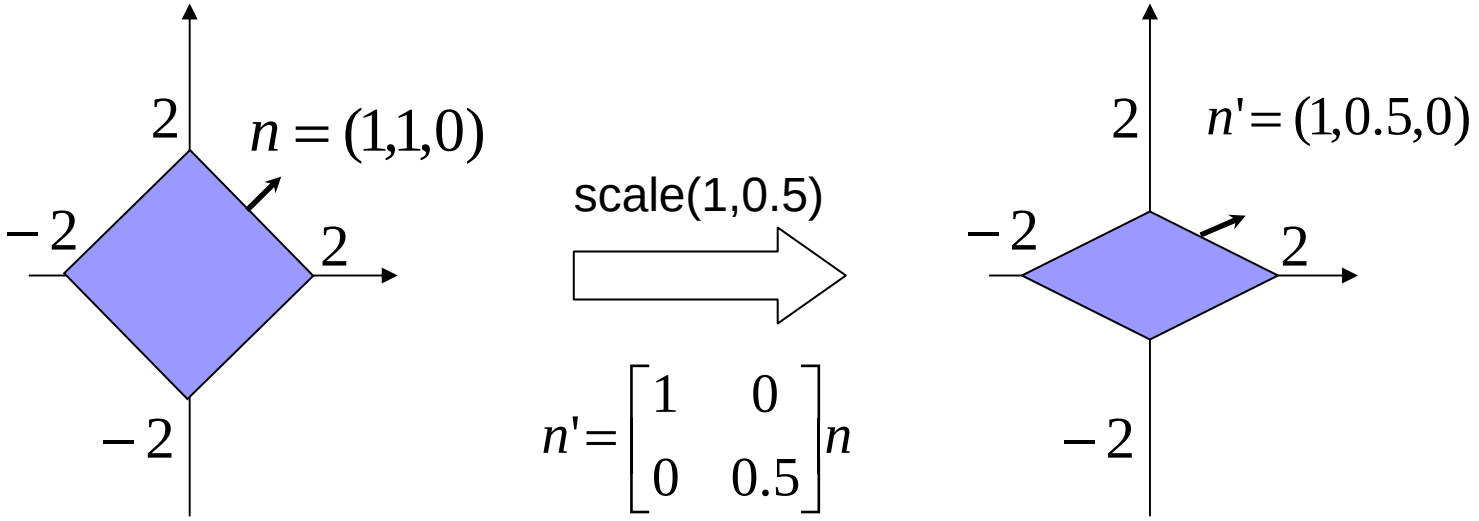
Lighting: trasformazione posizioni luce

- Le fonti di luce sono solidali con il punto di vista o con la scena? Ovvero….
- La posizione delle luci è nel frame di vista o in coordinate mondo?
- Entrambe utili

- La posizione della luce viene trasformata dalla matrice modelview_matrix corrente quando viene fatta la chiamata Light(…), proprio come fosse un punto (esempio)

Lighting: trasformazione delle normali (1)

- Le normali vengono specificate in coordinate mondo
- Assumiamo che quando di specificano siano effettivamente perpendicolari alla superficie
- È vero anche dopo la trasformazione???



Lighting: trasformazione delle normali (2)

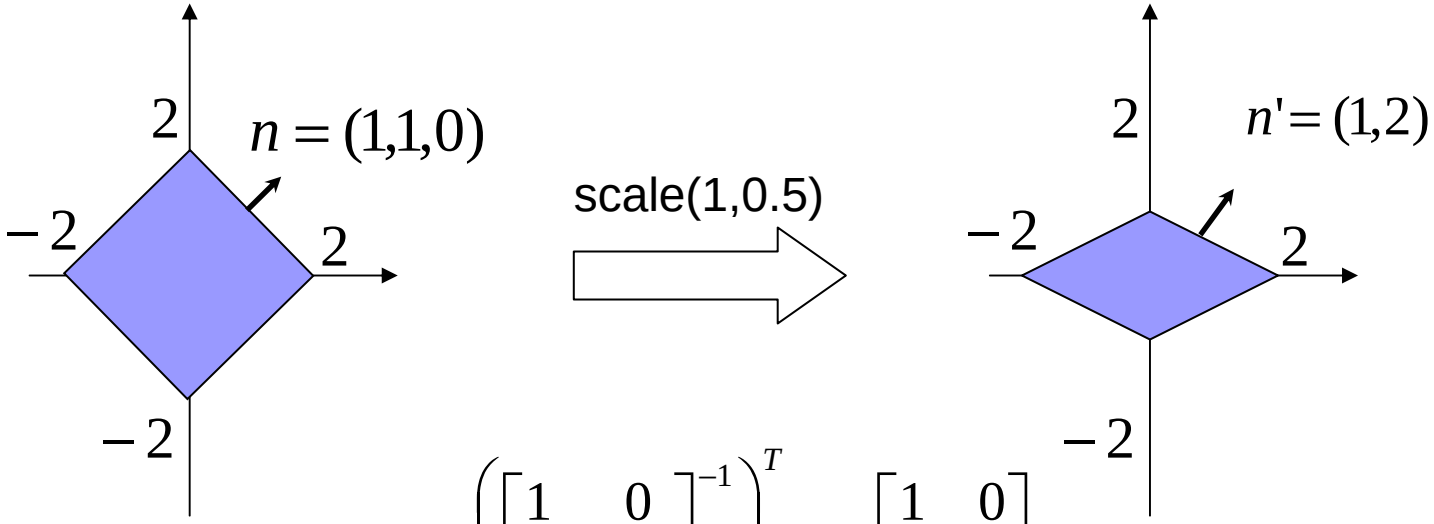
- Soluzione. Sia R la sottomatrice 3x3 più in alto a sinistra della matrice di modello (modelview_matrix)

$$\text{modelview\_matrix} = \begin{pmatrix} R & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix}$$

- Le posizioni dei vertici (o luci) vengono moltiplicate per modelview_matrix
- Le **normali** vengono moltiplicate per $\left(R^{-1}\right)^T$
- Proviamo sull'esempio…

Lighting: trasformazione delle normali (3)

- Funziona!



$$n' = \left(\begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}^{-1} \right)^{T} n = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} n$$

Texturing

Da Softogl.h

```
/// Filtraggio: NEAREST, BILINEAR
enum FILTERMODE {NEAREST,BILINEAR} ;

struct Texture{
        /// inizio dei dati dell'immagine
        unsigned char* begin;
        /// dimensioni
        int width, height;
};
/// vettore di tutte le texures allocate
extern std::vector<Texture> textures;

/// puntatore alla texture corrente. La texture corrente è quella acceduta da
/// Sample(float u, float v);
extern Texture * current_texture;

/// valore booleano per decidere se eseguire o no il texturing  (default a false)
extern bool isTexuringEnabled;

/// Imposta il valore della texture corrente current_texture
void BindTexture(int i);

int TexImage2D(unsigned char* tex,int width, int height);

Color4 Sample(float u, float v);
```

Texturing: caricare un’immagine da disco

- Non fa parte di softogl (ne’ di OpenGL)
- Usiamo le sdl:

SDL_Surface ***SDL_LoadBMP**(const char *file);

```
typedef struct SDL_Surface {
Uint32 flags; /* Read-only */
SDL_PixelFormat *format; /* Read-only */
int w, h; /* Read-only */
Uint16 pitch; /* Read-only */
void *pixels; /* Read-write */
 /* clipping information */
 SDL_Rect clip_rect; /* Read-only */
 /* Reference count -- used when freeing surface */
int refcount; /* Read-mostly */ /* This structure also contains private fields not shown here */

} SDL_Surface;
```