Fondamenti di Computer Graphics LM

Lab 7 - SHADER: Texture mapping, Normal mapping, Environmental & Refraction

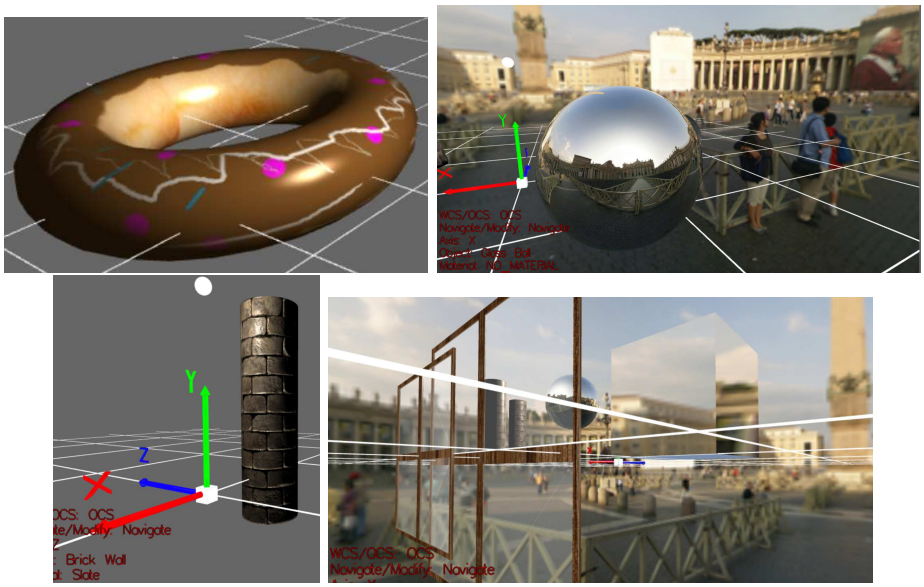


Figure 1: (top) Texture mapping 2D + Shading sul toro; effetto environment mapping sulla superficie sfera; (bottom) normal mapping sulla colonna; effetto trasparenza delle due finestre.

L'archivio file fornito contiene un semplice programma per la gestione di texture mapping, normal mapping e environment mapping con OpenGL e GLSL, compilare ed eseguire il programma fornito. Il programma è basato sul codice dell'esercitazione 3.

I tasti frecce dx e sx permettono di selezionare uno tra i seguenti modelli:

- mesh plane con 2D texture image creata in modo procedurale (checkboard) ai vertici;
- mesh cubo con 2D texture image caricata da file e associata alle facce.
- mesh toro creato attraverso la tassellazione di una superficie parametrica su un dominio parametrico. Con i tasti W/w e N/n si incrementa/decrementa il numero di avvolgimenti W/w e il numero di vertici per ogni avvolgimento N/n del raggio maggiore e del raggio minore.
- mesh sfera smooth (file .obj) creata con normali ai vertici e parametrizzazione associata ai vertici.
- mesh cilindro smooth (file .obj) con 2D texture image caricata da file.

Estendere il programma inserendo la gestione, mediante opportuni vertex e fragment shader, delle seguenti funzionalità:

1. **Camera motion** Permettere il movimento della camera virtuale lungo un percorso (curva) in modalitá look at (mediante curva chiusa di Bézier) in modo che la camera si muova lungo un percorso con centro di interesse un oggetto in scena.

2. **Texture mapping 2D** del toro con immagine letta da file di formato *nomefile.jpg* mediante gli shaders forniti *v_texture.glsl* e *f_texture.glsl*; si devono associare le coordinate texture (θ, ϕ) ad ogni vertice.

3. **Texture mapping 2D + Shading** Realizzare gli shaders *v_texture_phong.glsl* e *f_texture_phong.glsl* per combinare l'effetto shading Phong con la texture image sulla mesh toro; effetto illustrato in Fig. 1 sinistra.

4. **Procedural mapping** basato su un procedimento algoritmico a piacere sul toro.

5. **Normal mapping** di un oggetto mesh (column.obj o sharprockfree.obj) con normal map e immagine texture lette da file, mediante gli shaders *v_normal_map.glsl* (fornito) e *f_normal_map.glsl* (da creare sulla falsa riga di *f_phong.glsl* del LAB 3) utilizzando multitexturing GL_TEXTURE0 per DiffuseMap e GL_TEXTURE1 per NormalMap. L'effetto è illustrato in Fig. 1 (bottom-left). Confrontare l'effetto di normal mapping versus semplice texture mapping utilizzando lo shader TEXTURE_PHONG del punto 2. con la DiffuseMap come texture.

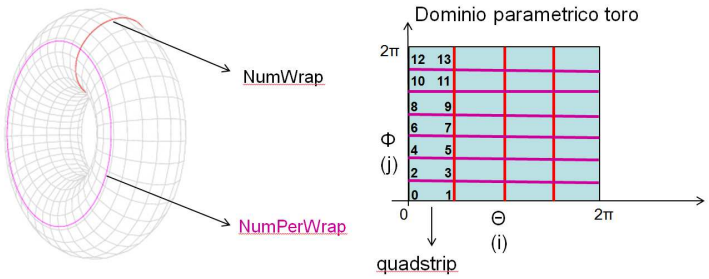
6. **Environment cube mapping: skybox** Gli shaders *v_skybox.glsl* e *f_skybox.glsl* realizzano l'effetto di resa dell'ambiente circostante mappato su un cubo che contiene la scena. Gestire il posizionamento e dimensionamento del cubo nell'applicazione affinché sia visibile l'effetto env map dell'ambiente di background come illustrato in Fig. 1 destra.

7. **Environment mapping: object REFLECTION** Realizzare gli shaders *v_reflection.glsl* e *f_reflection.glsl* per combinare l'effetto di resa dell'ambiente circostante su un oggetto riflettente in scena (es. sfera). Utilizzare come texture image il cube mapping (*cubeTexture*) come superficie intermedia; l'effetto sulla sfera é illustrato in Fig. 1 destra.

8. **Environment mapping: object REFRACTION** Analogamente a quanto fatto per il punto precedente, gestire un oggetto trasparente in scena (es. cubo). Realizzare gli shaders *v_refraction.glsl* e *f_refraction.glsl*.

9. OPZIONALE: **Oggetti semi-trasparenti: gestire in scena un paio di oggetti semi-trasparenti (es. window.obj) facendone la resa in drawScene() dal piú lontano al piú vicino dopo aver reso tutti gli oggetti opachi.**

Osservazione: Tassellazione e parametrizzazione della superficie toro Si ricorda che il toro ha la seguente rappresentazione parametrica $\mathbf{S}(\theta, \phi)$ con $\theta, \phi \in [0; 2\pi]$:



$$
\begin{aligned}
x(\theta, \phi) &= \sin(\theta)(R + r\cos(\phi)) \\
y(\theta, \phi) &= \sin(\phi) r \\
z(\theta, \phi) &= \cos(\theta)(R + r\cos(\phi))
\end{aligned}
$$

Il vettore normale è definito come

$$\mathbf{n}(\theta,\phi) \quad = \quad \mathbf{S}_\theta(\theta,\phi) \times \mathbf{S}_\phi(\theta,\phi)$$

con derivate parziali

$$\mathbf{S}_\theta(\theta,\phi) = \begin{bmatrix} dx/d\theta \\ dy/d\theta \\ dz/d\theta \end{bmatrix} \quad = \quad \begin{bmatrix} \cos(\theta)(R + r\cos(\phi)) \\ 0 \\ -\sin(\theta)(R + r\cos(\phi)) \end{bmatrix}$$

$$\mathbf{S}_\phi(\theta,\phi) = \begin{bmatrix} dx/d\phi \\ dy/d\phi \\ dz/d\phi \end{bmatrix} \quad = \quad \begin{bmatrix} -r\sin(\phi)\sin(\theta) \\ r\cos(\phi) \\ -r\sin(\phi)\cos(\theta) \end{bmatrix}$$