LAB 5 – Texture Mapping

Fondamenti di Computer Graphics M
Erika Gardini – Ingegneria Informatica A.A. 2016/2017

Per prima cosa è stato scaricato, compilato ed eseguito il programma di partenza per la gestione di texture 2D con OpenGL.
Una volta sperimentate le funzionalità di base è stato possibile passare alle richieste dell'esercitazione.
In particolare, si richiedeva di realizzare le seguenti funzionalità:

- permettere il texture mapping 2D del toro con immagini lette da file di formato nomefile.bmp;
- permettere l'environment mapping sferico/cubico sfruttando il two-step mapping in modalità OpenGL;
- permettere il procedural mapping basato su un procedimento algoritmico a piacere.

Parte 1: texture mapping 2D

Per realizzare questa funzionalità è stato necessario, come prima cosa, caricare le texture in formato .bmp. Una volta indicati i percorsi dei file, il metodo `initFour` già fornito consentiva il caricamento delle texture.
Una volta caricate le texture è stato necessario eseguire i seguenti comandi in modalità torus:

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, textureName[0]);
```

In particolare, il primo comando abilita la modalità texture ed il secondo consente di indicare quale texture si desidera associare al toro fra quelle caricate.
A questo punto è necessario "agganciare" i punti della texture 2D a quelli del toro. Per far questo occorre sfruttare la parametrizzazione. Questa viene effettuata mediante il metodo `putVert`, il quale associa ai punti del toro i punti della texture.
In particolare, l'algoritmo utilizzato è il seguente:

```
for (i=0; i<NumWraps; i++) {
     glBegin(QuadMode==1 ? GL_QUAD_STRIP : GL_TRIANGLE_STRIP);
     for (j=0; j<=NumPerWrap; j++) {
          putVert(i, j);
          putVert(i+1, j);
     }
    glEnd();
}
```

dove,

```
void putVert(int i, int j) {

    float phi = PI2*j/NumPerWrap;
    float theta = PI2*i/NumWraps;
    float sinphi = sin(phi);
    float cosphi = cos(phi);
    float sintheta = sin(theta);
    float costheta = cos(theta);
    float r = MajorRadius + MinorRadius*cosphi;

    // the normal is the cross product of the partial derivatives
    glNormal3f(sintheta*cosphi, sinphi, costheta*cosphi);

    int numTexVer = 2;
    int numTexHor = 2;
```

```
    float verticalTexRate = j / (float)NumPerWrap;
    float horizontalTexRate = i / (float)NumWraps;

    glTexCoord2f(verticalTexRate*numTexVer,horizontalTexRate*numTexHor);

    // place the vertex
    glVertex3f(sintheta*r, MinorRadius*sinphi, costheta*r);

}
```

Quindi, se per esempio si avesse

$$numTexVert = numTexHor = 2$$
$$numWraps = numPerWrap = 4$$

si otterrebbe:

i	*j*	*putVert(i,j)*	*glTextCoord2f*	*putVert(i+1,j)*	*glTextCoord2f*	*Index Image*
0	0	putVert(0,0)	$\frac{0}{4}\cdot 2 = 0, \frac{0}{4}\cdot 2 = 0$	putVert(1,0)	$\frac{0}{4}\cdot 2 = 0, \frac{1}{4}\cdot 2 = 0.5$	1,2
0	1	putVert(0,1)	$\frac{1}{4}\cdot 2 = 0.5, \frac{0}{4}\cdot 2 = 0$	putVert(1,1)	$\frac{1}{4}\cdot 2 = 0.5, \frac{1}{4}\cdot 2 = 0.5$	3,4
0	2	putVert(0,2)	$\frac{2}{4}\cdot 2 = 1, \frac{0}{4}\cdot 2 = 0$	putVert(1,2)	$\frac{2}{4}\cdot 2 = 1, \frac{1}{4}\cdot 2 = 0.5$	-
0	3	putVert(0,3)	$\frac{3}{4}\cdot 2 = 1.5, \frac{0}{4}\cdot 2 = 0$	putVert(1,3)	$\frac{3}{4}\cdot 2 = 1.5, \frac{1}{4}\cdot 2 = 0.5$	-
…	…	…	…	…	…	…

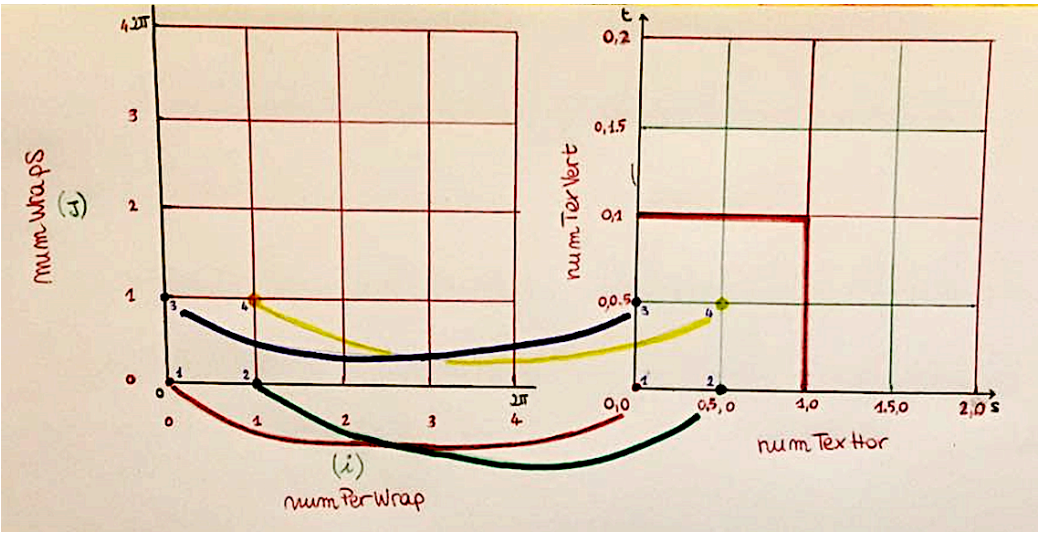


Figura 1: parametrizzazione

Infine, terminato l'algoritmo, si disabilita la modalità texture con il seguente comando:

```
glDisable(GL_TEXTURE_2D);
```

Eseguendo il programma si ottiene il seguente risultato:

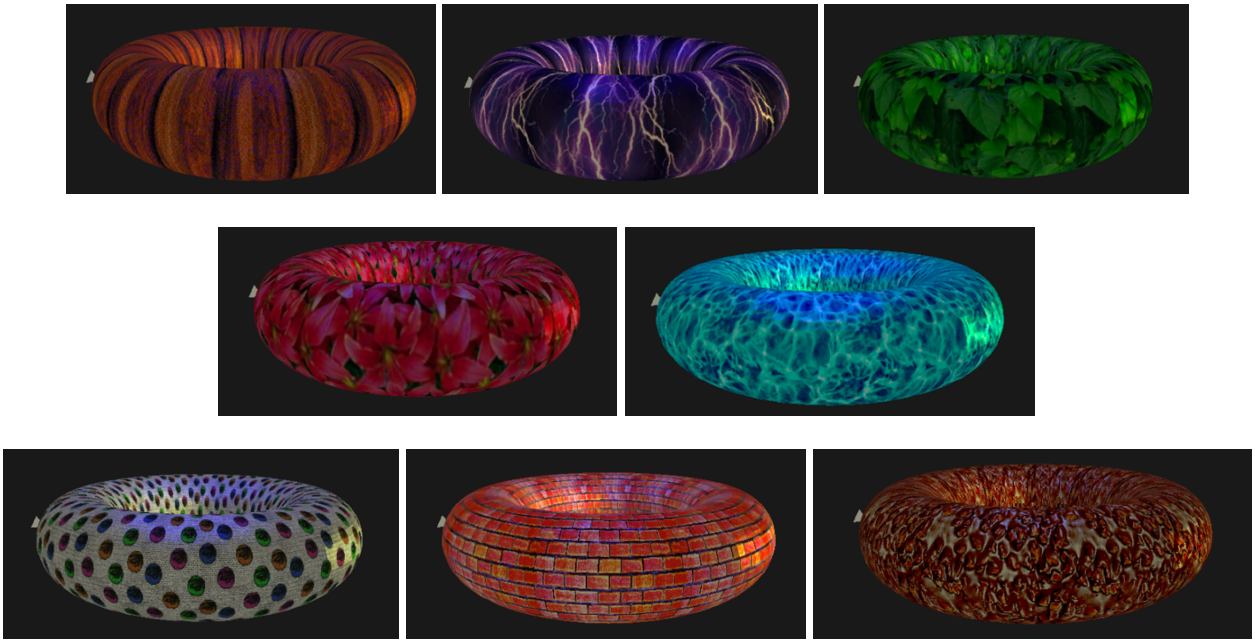


Figura 2: texture mapping

Parte 2: environmental mapping

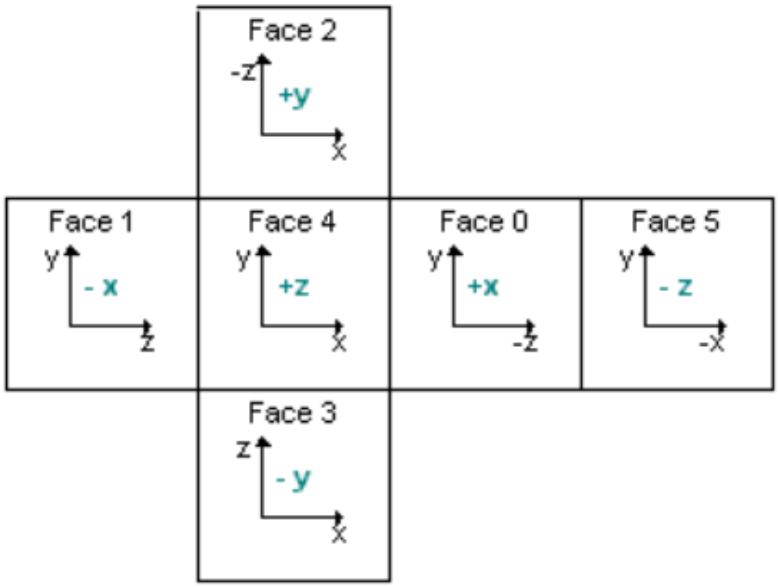
L'environmental mapping simula la riflessione degli oggetti in scena su di un oggetto riflettente (nel caso dell'esercitazione il toro).
La texture dovrebbe contenere una vista del mondo avente come punto di interesse l'oggetto.
Per prima cosa è necessario creare la texture; per farlo è necessario effettuare un rendering della scena senza l'oggetto riflettente (il toro) con la camera al centro dell'oggetto diretta verso la normale all'oggetto stesso.
A seconda dell'environment (cubico o sferico), le texture possono avere i seguenti aspetti:

Figura 3: texture sferica o cubica

Una volta creata la texture è necessario effettuare nuovamente il rendering utilizzando la texture mapping per apportare i contributi di riflessione nella superficie riflettente dell'oggetto (il toro).

Per realizzare le funzionalità dell'esercitazione è stato sufficiente effettuare solo il secondo passo, in quanto le texture (environmental map) erano già fornite.

Environmental mapping cubico
Per realizzare questa funzionalità è stato necessario caricare le immagini già fornite seguendo lo schema di seguito riportato



ed utilizzando il seguente comando:

```
glTexImage2D(“TEXTURE_TARGETS”,0,GL_RGB,256,256,0,GL_RGB,
GL_UNSIGNED_BYTE, “bitmap_data”);
```

Il comando è stato ripetuto sei volte, ciascuna delle quali con uno dei seguenti parametri “texture targets”:

```
GL_TEXTURE_CUBE_MAP_POSITIVE_X
GL_TEXTURE_CUBE_MAP_NEGATIVE_X
GL_TEXTURE_CUBE_MAP_POSITIVE_Y
GL_TEXTURE_CUBE_MAP_NEGATIVE_Y
GL_TEXTURE_CUBE_MAP_POSITIVE_Z
GL_TEXTURE_CUBE_MAP_NEGATIVE_Z
```

e con la corrispondente immagine, indicata nel parametro “bitmap_data”.
Dopo aver creato la texture cubica è necessario creare l’oggetto texture ed associarlo all’oggetto della scena (il toro); per far questo è necessario utilizzare i seguenti comandi:

```
glGenTextures(1, &tex_cubo);
glBindTexture(GL_TEXTURE_CUBE_MAP, tex_cubo);
```

Infine, è stato necessario invocare i seguenti comandi per la generazione automatica delle coordinate della texture map cubica:

```
glEnable(GL_TEXTURE_CUBE_MAP);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);
```

Prima di eseguire l'applicazione sono stati inseriti alcuni filtri, in modo tale da migliorare il texturing. In particolare, sono stati eseguiti i seguenti comandi:

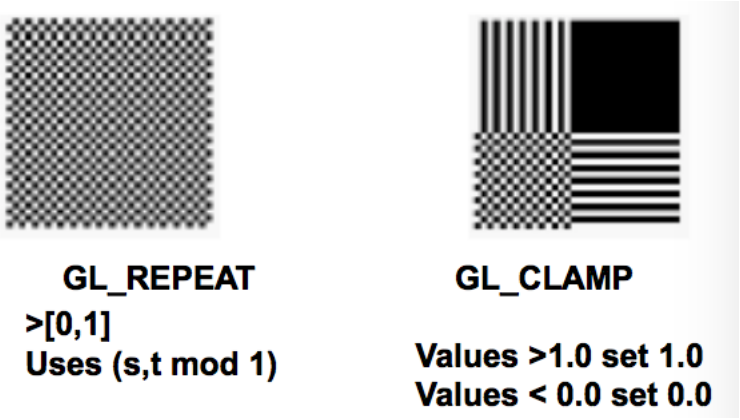
```
glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_S,GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_T,GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_CUBE_MAP,GL_TEXTURE_WRAP_R,GL_CLAMP_TO_EDGE);
```

I primi due consentono di ingrandire o ridurre la texture quando le dimensioni di un textel non coincidono con la dimensione di un pixel.
Nel caso di ingrandimento (magnification) occorre calcolare il colore nel nuovo spazio generato, o attraverso l'interpolazione oppure utilizzando il colore del pixel (costante).
Nel caso di riduzione (minification), invece, occorre sintetizzare il colore in un solo pixel.
L'ultimo parametro del filtro consente di scegliere come calcolare il colore e può assumere i seguenti valori:

- `GL_LINEAR`
- `GL_NEAREST`
- `GL_LINEAR_MIPMAP_LINEAR`

Gli ultimi tre filtri consentono di riportare le dimensioni della texture nel range [0,1]x[0,1]; il filtro è applicato per ciascuna coordinata (s, t, r).
L'ultimo parametro ammette i seguenti valori:

- `GL_REPEAT`
- `GL_CLAMP`



A questo punto è stato eseguito il programma. I risultati ottenuti sono i seguenti:

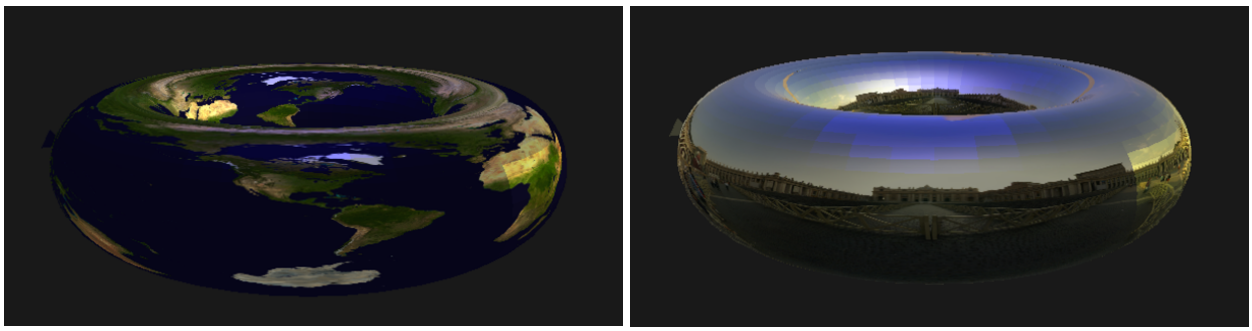


Figura 4: environmental mapping cubico

Environmental mapping sferico
Per realizzare l'environmental mapping sferico, invece, il procedimento è molto simile a quello seguito per il texture mapping (in quanto l'environmental texture era già fornita).
In particolare, è necessario:

- abilitare la modalità texture:

```
glEnable(GL_TEXTURE_2D);
```

- generare le coordinate della texture map sferica (così come per l'environmental texture cubico):

```
glTexGeni(GL_S,GL_TEXTURE_GEN_MODE ,GL_SPHERE_MAP);
glTexGeni(GL_T,GL_TEXTURE_GEN_MODE ,GL_SPHERE_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

- selezione della texture da associare all'oggetto (il toro):

```
glBindTexture(GL_TEXTURE_2D, sphereName);
```

Qui di seguito si riportano i risultati ottenuti:

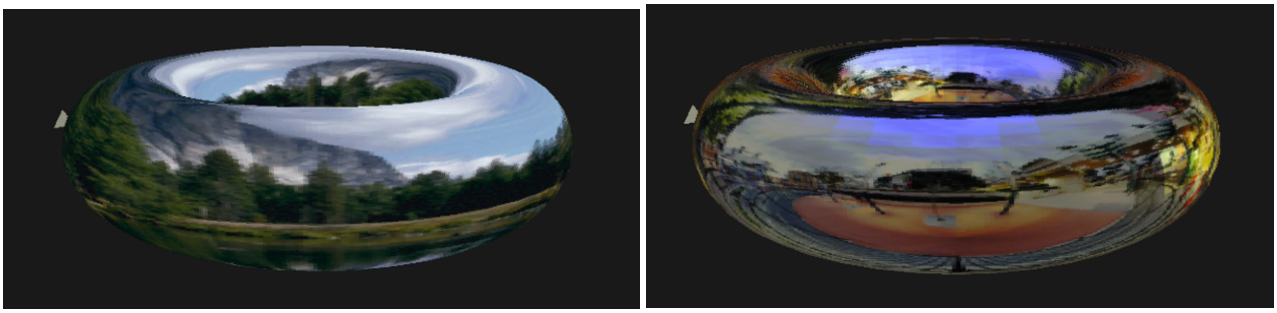


Figura 5: environmental mapping sferico

In entrambi gli environmental mapping, dopo aver selezionato la texture da associare all'oggetto, viene invocato il metodo `putVert` che associa effettivamente i punti della texture all'oggetto stesso.

Parte 3: procedural mapping basato su un procedimento algoritmico a piacere
Per realizzare questa modalità, dopo aver attivato la modalità texture (mediante lo stesso comando della parte 1) è stato invocato il seguente metodo `initCheckerTextures()` il quale contiene la logica per la realizzazione del procedural mapping.
A tal scopo si è deciso di realizzare una texture da applicare al toro che fosse costituita da righe orizzontali di tre colori alternati: rosso, verde e blu.
A questo punto è stato sufficiente proseguire con lo stesso procedimento della parte 1, ossia "agganciando" i punti della texture 2D a quelli del toro, sempre mediante il metodo `putVert`.

Anche in questo caso sono stati applicati dei filtri, i quali vengono riportati qui di seguito:

```
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
```

Il significato di ciascuno di essi è equivalente a quello indicato nell'environmental mapping cubico.

Il risultato ottenuto è il seguente:

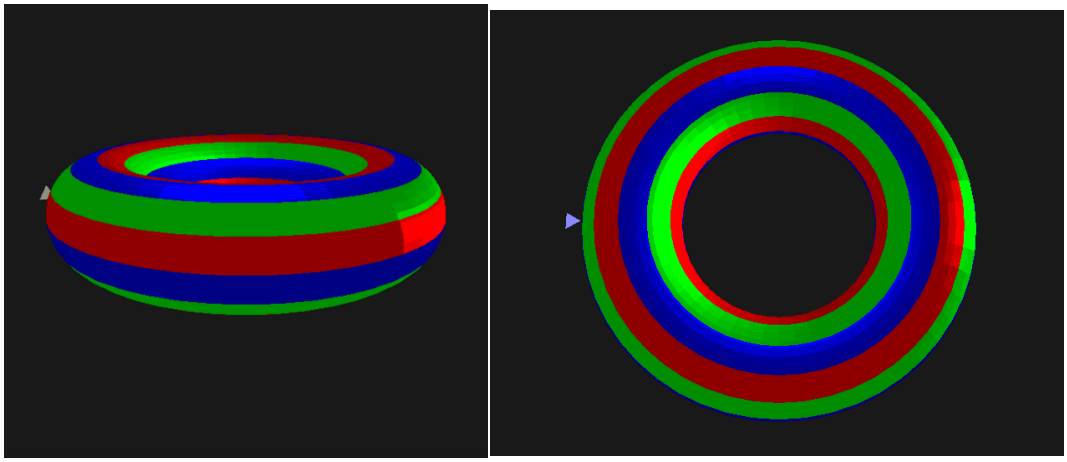


Figura 6: procedural mapping