Fondamenti di Computer Graphics M
Report LAB 03
Navigazione interattiva in scena con modelli mesh 3D

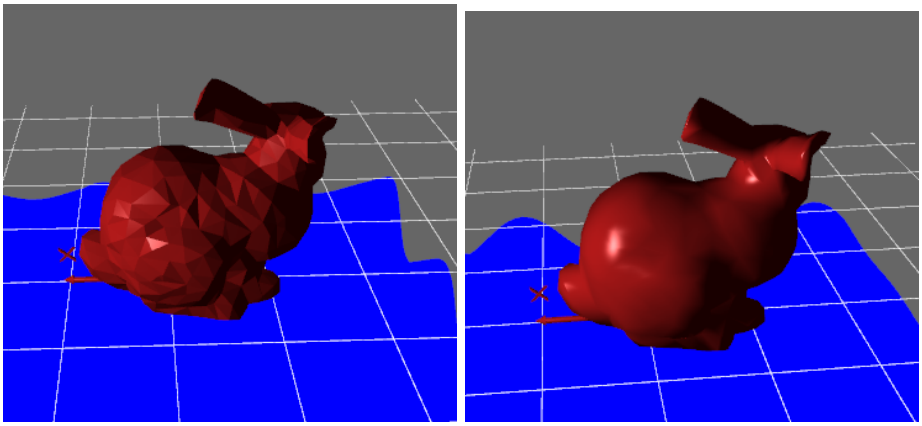
Giulio Posati

June 2, 2023

Indice

1 Calcolo e memorizzazione delle normali ai vertici per i modelli mesh poligonali. Visualizzazione in modalita normali alle facce (flat shading) e normali ai vertici (sia GOURAUD shading sia PHONG shading)

Quando un oggetto viene caricato all'interno della nostra applicazione è possibile salvare le sue normali sia relativamente alle sue facce oppure prendendo in considerazione le normali ai vertici, semplicemente andandolo a specificare all'interno della funzione "loadObjFile". In base a questa scelta otterremo una visualizzazione diversa dell'oggetto stesso.



(a) Normali alle facce (b) Normali ai vertici

Figure 1: Caricamento delle normali

2 Provare a creare un materiale diverso da quelli forniti

Per la creazione di un nuovo materiale, innanzitutto sono state create le sue componenti di colore:

```
glm::vec3 my_material_ambient = { 0.1, 0.1, 0.1 };
glm::vec3 my_material_diffuse = { 0.5, 0.2, 0.5 };
glm::vec3 my_material_specular = { 0.88, 0.08, 0.8 };
GLfloat my_material_shininess = 100.0f;
```

Poi è stato creato un nuovo materiale a cui sono state associate le proprietà appena definite, all'interno della funzione "init()":

```
materials[MaterialType::MY_MATERIAL].name = "MyMaterial";
```

```
materials[MaterialType::MY_MATERIAL].ambient = my_material_ambient;
materials[MaterialType::MY_MATERIAL].diffuse = my_material_diffuse;
materials[MaterialType::MY_MATERIAL].specular = my_material_specular;
materials[MaterialType::MY_MATERIAL].shininess = my_material_shininess;
```

Fatto questo è sufficiente assegnare il nuovo materiale ad un oggetto presente in scena all'interno della sua funziona "init".

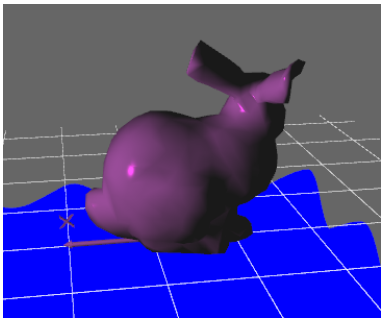


Figure 2: Creazione nuovo materiale

3 Wave motion

La realizzazione di questa operazione richiede la creazione di un apposito vertex shader in grado di controllare la coordinata y dei vertici di un piano, tramite una variabile che rappresenta lo scorrere del tempo, passata da applicazione. L'applicazione all'interno della funzione "initShader()" caricherà lo shader in modo simile a quanto già fa per gli altri tipi di shading (il colore viene semplicemente impostato a blu). Infine, all'interno della funzione "drawScene()" occorre invece specificare la variabile che rappresenterà lo scorrere del tempo.

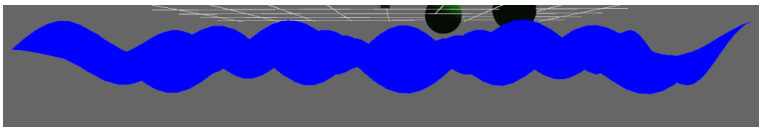


Figure 3: Wave motion

4 Toon shading

Anche per questa operazione occorre realizzare nuovi shader: viene introdotta solamente l'implementazione di "f_toon.glsl" poiché "v_toon.glsl" è realizzato come "v_blinn.glsl": l'idea dietro al toon shading è quella di colorare le parti di un'oggetto in modo proporzionale all'intensità della luce che lo colpisce, ma discretizzando le sfumature di colore in un piccolo intervallo (in questo caso

abbiamo 5 diverse tonalità di blu). Non viene riportato il codice relativo al setup di questo tipo di shading in quanto coerente con l'utilizzo degli shader forniti.

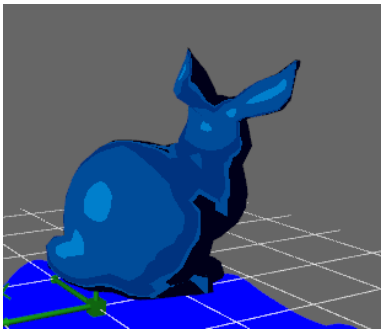


Figure 4: Toon shading

5 Scorrimento orizzontale della camera

Per permettere lo scorrimento orizzontale della camera, occorre traslare sia la posizione della camera che del suo punto di riferimento, in modo simile allo spostamento verticale (aggiungendo all'oggetto ViewSetup un vettore che specifica la direzione orizzontale). Per questo sono state implementate due funzioni, chiamate dall'applicazione dopo la pressione del tasto SHIFT, in base al movimento della rotella del mouse.

6 Trasformazione degli oggetti in scena

L'applicazione è stata fornita con il setup già pronto per l'utilizzo di funzioni di trasformazione sugli oggetti in scena. L'unica cosa da fare, pertanto è implementare la funzione "modifyModelMatrix()":

- Caso OCS: calcola la matrice di trasformazione e moltiplica la matrice dell'oggetto per questa.
- Caso WCS: stesso procedimento ma viene usata l'inversa della matrice dell'oggetto che dopo essere stata moltiplicata per la matrice di trasformazione viene invertita.