21 apr 2017

Fondamenti di Computer Graphics LM

Lab 4 - Ray tracing - Digital Art

Questa esercitazione può essere eseguita sia in ambiente Windows che Linux.

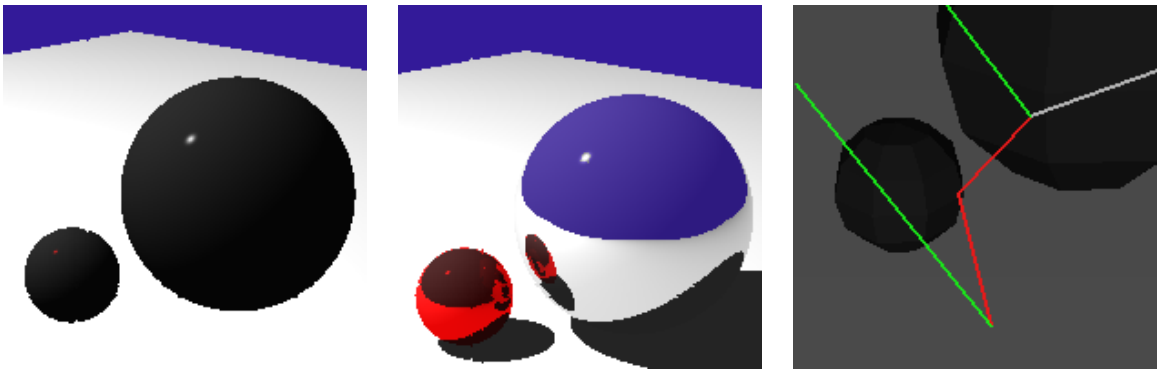
1. **Ray Tracing**

   Scaricare, compilare ed eseguire il programma fornito. Tramite il tasto 'r' è possibile avviare il raytracing dalla posizione corrente della camera. L'implementazione iniziale contiene solo la procedura di ray casting e l'obiettivo dell'esercitazione, una volta familiarizzato con l'ambiente ed esplorato le potenzialitá dell'algoritmo, è di:

   (a) sviluppare la gestione degli shadow rays per la generazione di hard shadows

   (b) sviluppare la gestione ricorsiva dei reflection rays.

   (c) *OPZIONALE:* Implementare un meccanismo per la gestione di soft shadows mediante risorse luminose ad area anzichè puntiformi. Nella versione attuale le luci in scena sono già area lights implementate come quads con emissione non-zero, ma se ne considera solo il baricentro per gestirle come puntiformi.



a) b) c)

Figure 1: (a) l'output dell'applicazione con rendering tramite ray casting; (b) il rendering è eseguito tramite raytracing, dopo aver implementato la gestione degli shadow rays e dei reflection rays. (c) la modalita' di debug con visualizzazione dei raggi generati

L'applicazione offre una modalità di debug che permette di visualizzare i vari raggi in scena. Alla pressione del tasto 't' un raggio passante per il pixel correntemente sotto il puntatore del mouse è tracciato nella scena e visualizzato.

L'esecuzione dell'applicazione necessita la specifica di argomenti dalla linea di comando. Ad esempio:

```
raytracing.exe -size 600 600 -background_color 0.1 0.1 0.1 -num_shadow_samples 10 -num_bounces 3 -input reflective_spheres.obj
raytracing.exe -size 600 600 -background_color 0.1 0.1 0.1 -num_shadow_samples 10 -num_bounces 3 -input cornell_box_reflective_spheres.obj
```

Struttura dell'applicazione: devi mettere nella cartella Debug i file che usa.

- Codice di base (argparser.h, boundingbox.h, boundingbox.cpp, camera.h, camera.cpp, glCanvas.h, glCanvas.cpp, main.cpp, matrix.h, matrix.cpp, utils.h, vectors.h, Makefile)
- Half-Edge Data Structure (edge.h, edge.cpp, face.h, face.cpp, mesh.h, mesh.cpp, sphere.h, sphere.cpp, material.h, material.cpp, vertex.h, vertex_parent.h, bag.h)
- Raytracing (raytracer.cpp, raytracer.h, ray.h, hit.h, raytree.h, raytree.cpp)
- Scene in formato obj non standard (reflective_spheres.obj, cornell_box.obj, l.obj, cornell_box_diffuse_sphere.obj, cornell_box_reflective_spheres.obj)

2. **Digital Art: uso di Ray tracing in Blender**

   L'obiettivo è quello di utilizzare il rendering con ray tracing per creare almeno una scena (Digital Art) con oggetti di tipologie diverse (Bézier, NURBS, mesh, ...) sfruttando al meglio le potenzialità del ray tracing. Non è espressamente richiesto l'uso di texture, ma possono essere gestite, si richiede di curare particolarmente materiali, luci ed effetti speciali.

   Si richiede inoltre una breve relazione (3/4 pagine) di descrizione delle tecniche utilizzate e alcuni screenshot della scena resa con ray tracing. Si valuteranno le strategie utilizzate e gli effetti creati così come l'estetica di quanto prodotto sia questo terribilmente bello che accuratamente brutto. Buon divertimento!

   NOTA: L'utilizzo di Blender non è vincolante: lo stesso esercizio può essere svolto in altri ambienti di modellazione/grafica che supportino illuminazione globale con ray tracing (es. POVRAY).