Esercitazione 6

Stefano Romanazzi

Student ID: 819445

E-mail: stefan.romanazzi@gmail.com

Panoramica

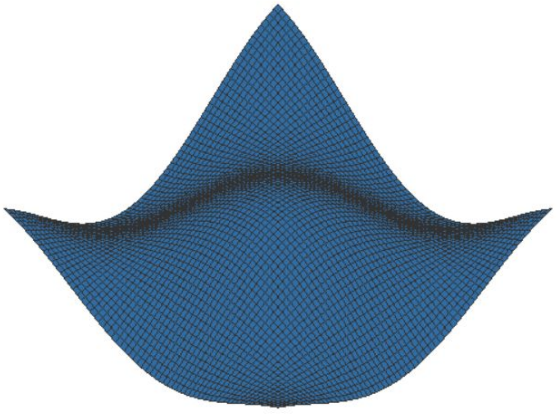
La presente esercitazione contiene svariate mini-esercitazioni indipendenti volte ad integrare l'utilizzo degli shader applicati a tecniche già viste nelle esercitazioni precedenti. Nei paragrafi successivi si andrà a trattare nel merito ogni modifica effettuata in ogni esercitazione.

Wave motion

Per creare l'effetto di ondeggiamento al foglio è bastato accedere al vertex shader modificando la coordinata y di un vettore con la seguente formula:

$$v_y = \alpha \sin(\omega t + 5v_x) \sin(\omega t + 5 v_z)$$

omega, *alfa* e *t*, sono variabili impostate nel file C++. Con il tasto sinistro del mouse è possibile modificare l'ampiezza dell'ondeggiamento (α), mentre con il tasto destro è modificata la frequenza. I valori ciclano tra valori preimpostati e la loro modifica nel file .cpp si ripercuote sulle corrispondenti variabili del vertex shader. In basso è fornita un'immagine che mostra l'ondeggiamento del foglio con ampiezza e frequenza massime.

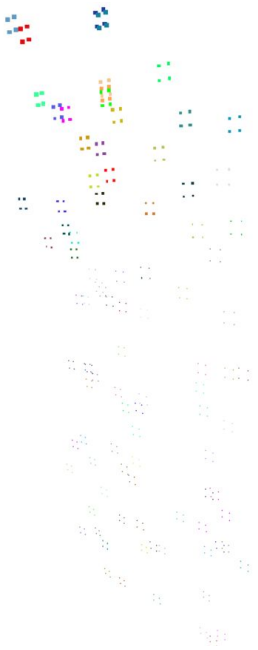


Particle system

Nell'esercitazione del particle system sono mostrate varie particelle randomizzate in movimento lungo l'asse x e y. Scopo dell'esercitazione è quello di manipolare la dimensione delle particelle in funzione della loro altezza e di far spostare le particelle anche lungo l'asse z. La modifica della dimensione dei punti tramite vertex shader è abilitata tramite le funzioni *glEnable(GL_POINT_SPRITE)* e *glEnable(GL_VERTEX_PROGRAM_POINT_SIZE)*. La dimensione

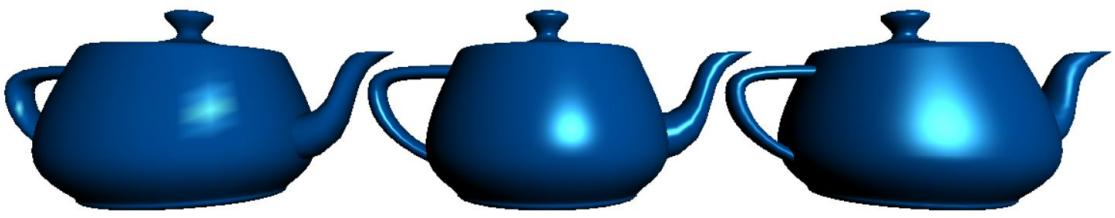
delle particelle è stata modificata nel vertex shader tramite l'attributo gl_pointsize, che varia dunque con la variazione della coordinata y.

Per aggiungere lo spostamento lungo l'asse z invece è stata mappata anche la coordinata v_z, mappata a partire dal codice C++. Una volta eseguito il mapping tramite la funzione si è proceduto con la variazione della coordinata z in funzione del tempo, esattamente come negli altri casi: "*t.z = gl_Vertex.z + v_z*time + 0.5 * a * time*" permette di raggiungere l'obiettivo. Di seguito è mostrata un'immagine che fa notare il cambio di dimensione dele particelle in funzione della loro y.



Phong Lighting

Nell'esercitazione del Phong Lighting è richiesta l'aggiunta di un terzo shader per calcolare il modello di illuminazione secondo la formula di Phong e calcolando anche la componente speculare. Gli shader già presenti si basano sul calcolo dell'half-way vector di Blinn e senza vettore di riflessione. Il terzo shader calcola dunque i coefficienti di Phong, in particolare il coefficiente K_s, tramite la formula float *K_s = pow(max(dot(R, N), 0.0), gl_FrontMaterial.shininess)*, dove R è il vettore di riflessione e N la normale al vertice.
Per eseguire un confronto dei tre modelli implementati è stata aggiunta la visualizzazione di una terza teapot così da visualizzare il risultato ottenuto.

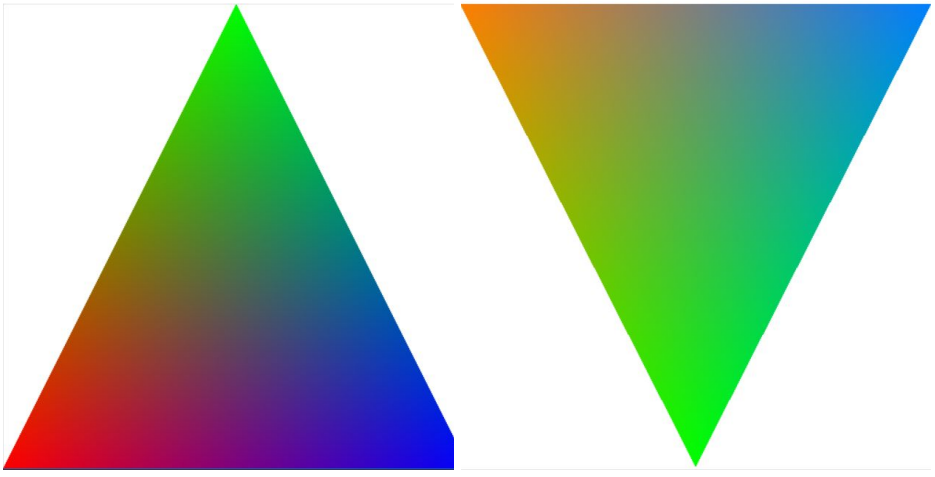


Toon Shading

Questa esercitazione fa riferimento al progetto NONPHOTO e ha come scopo quello di aggiungere un outline di colore diverso ai bordi della teiera. L'effetto *silhouette* è dunque aggiunto calcolando il l'ampiezza dell'angolo tra il vettore che parte dall'osservatore verso il vertice e la normale al vertice stesso. Se il dot product tra i due vettori restituisce un valore al di sotto di una certa soglia (0.3), allora il punto sarà colorato di rosso. Tali modifiche sono state effettuate modificando il pixel shader *f.glsl*. Infine è mostrato il risultato ottenuto dopo aver aggiunto l'effetto silouhette.

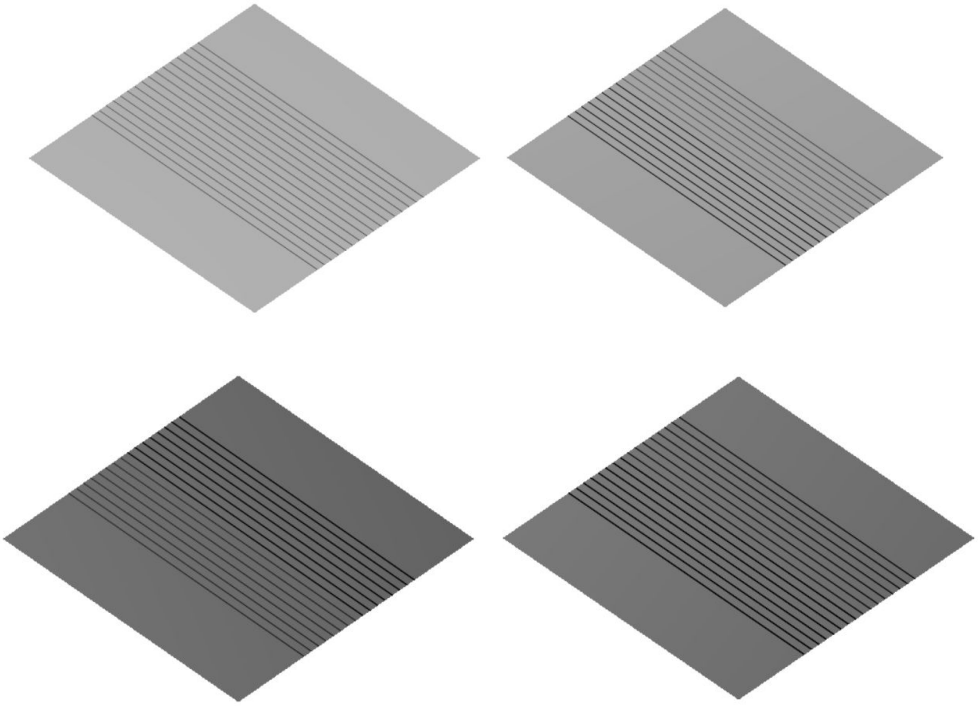
Morphing

Nell'esercitazione del morphing è presente un triangolo in movimento, realizzato mediante l'interpolazione dei colori di tre vertici. L'obiettivo dell'esercitazione è quello di modificare il colore dei vertici al variare del loro movimento. La posizione dei vertici è alterata mediante la funzione *mix* che consente di ricalcolare la posizione dei vertici mediante la funzione oscillatoria s. Al variare del movimento dei vertici è dunque richiesto un cambio del loro colore. È stato deciso di implementare questo effetto modificando il colore dei vertici in funzione del valore della funzione oscillatoria, pertanto ricalcolando *gl_FrontColor* come *gl_FrontColor = gl_Color + vec4(0, s/2, 0, 1),* dove s è la funzione oscillatoria. Sommando le nuove componenti ai colori preesistenti è stato possibile ottenere l'effetto mostrato nella figura sottostante.



Bump mapping

Nell'esercitazione Bump mapping è rappresentata una superficie illuminata mediante il modello di illuminazione creato negli shader, con una luce mobile che al variare della sua posizione evidenzia l'height field creato nel codice C++. Lo scopo di tale esercitazione è quello di modificare height field fornito in input, così da modificare l'effetto visivo ottenuto. L'obiettivo è stato raggiunto ricreando l'height field, popolandolo con normali identiche su tutta la superficie, fatta eccezione per 8 linee parallele di normali aventi valori diversi sulle coordinate. 4 di queste linee hanno come valori delle normali [0.5, 0.5, 0.0], mentre le altre 4 hanno come valori [0.5, 0.0, 0.0]. L'impostazione di tali valori consente di visualizzare soltanto parte delle linee aggiunte al variare dello spostamento della luce. Di seguito è mostrato l'effetto visivo ottenuto in istanti diversi durante l'esecuzione del programma.



Cube environment mapping

Nell'ultima esercitazione è invece richiesto di modificare la texture associata ad una teapot tramite cube mapping. Inizialmente è mostrata la teapot con sei colori differenti. La texture da applicare alla teapot proviene dall'esercitazione 5 e rappresenta una mappa geografica. Trattandosi di cube mapping, questa texture è stata selezionata in quanto è già suddivisa in 6 file differenti e il ciò la rende adatta ad essere utilizzata nell'esercitazione. Viene dunque caricata la texture tramite la libreria *RgbImage*, fornita nel progetto e memorizzata all'interno del vettore textureData, contenente per l'appunto sei elementi, uno per ogni texture. È poi eseguito il cube mapping, tramite la funzione *glTexImage2D*, che associa ciascuna texture al lato corretto del cubo. Tale texture è infine applicata alla teapot tramite la funzione *glBindTexture*. La visualizzazione della texture inserita è attivabile tramite la pressione del pulsante 'a'. L'effetto ottenuto è dunque mostrato qui di seguito.