

# GLSL

## Descrizione

Il programma ha come scopo quello di mostrare alcuni possibili utilizzi degli shader scritti in GLSL (OpenGL Shading Language)

## Obiettivi

1. Wave Motion
2. Particle System
3. Phong Lighting
4. Toon Shading
5. Morphing
6. Bump Mapping
7. Cube Environment Mapping

## Risultati

### 1. Wave Motion

In questo esercizio si animano i vertici di una mesh quadrata, usando un vertex shader per simulare il movimento di un'onda. L'altezza dei vertici viene perturbata sfruttando la seguente formula basata sulla funzione seno:

$$v.y = A * \sin(\omega * \text{time} + 5.0 * v.x) * \sin(\omega * \text{time} + 5.0 * v.z);$$

I 3 parametri passati allo shader sono di tipo **uniform** e sono:

- **time**, il tempo (time)
- **A**, l'ampiezza dell'onda, modificabile tramite click sinistro fra i valori 0.05, 0.1 e 0.2
- **omega**, la frequenza, modificabile tramite click destro fra i valori 0.0005, 0.001 e 0.002

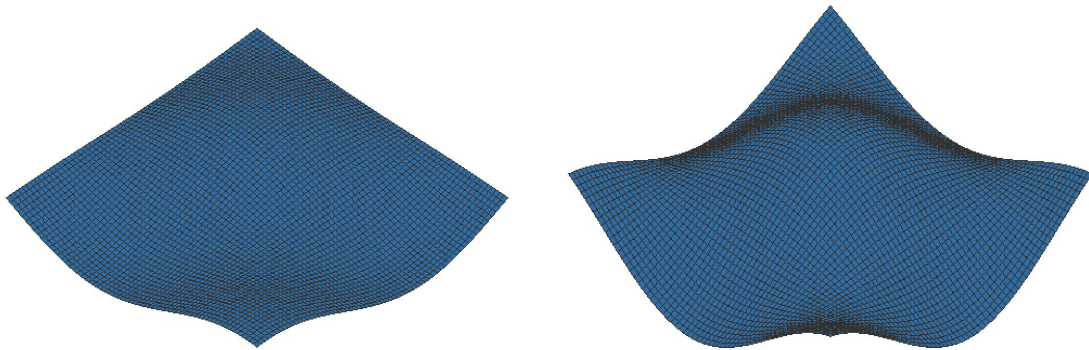


Fig 1. wave motion con ampiezza 0.05 (a) e 0.2 (b)

## 2. Particle System

In questo esercizio si implementa un sistema particellare definendo posizione, dimensione e colore di ogni particella all'interno di un vertex shader.

Per fare sì che la **dimensione della particella dipenda dall'altezza** a cui si trova, è stato usato `GL_POINTS` per definire il tipo di primitiva, e la formula

$$\text{gl\_PointSize} = 2.0 * (5.0 + t.y)$$

che tramite `gl_PointSize` imposta la grandezza di ogni punto di un valore multiplo della coordinata `y`.

È stato accentuato il **movimento delle particelle sull'asse z** utilizzando la formula

$$t.z = \text{gl\_Vertex}.z + (vz * 9.0 * \text{time});$$

È stato inoltre aggiunto un effetto "spegnimento" delle particelle con la seguente formula

$$\text{gl\_FrontColor} = \text{gl\_Color} * t;$$

che ha come effetto quello di rendere più scure le particelle al passare del tempo.

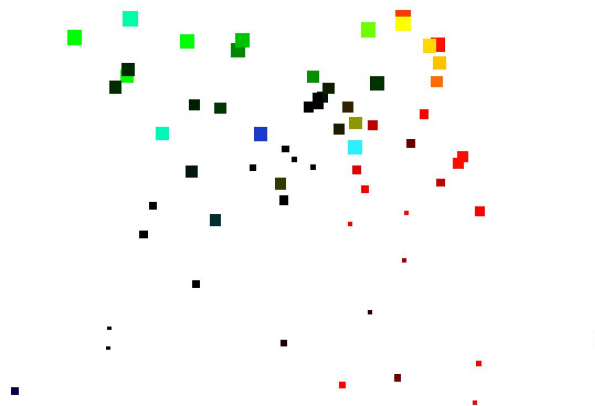


Figura 2. Particle System ottenuto

### 3. Phong Lighting

L'esercitazione mostra come la tecnica di implementazione di Phong influisca sulla resa finale del rendering. A copie diverse della stessa mesh vengono applicati 3 shader per la valutazione del modello di illuminazione:

1. interpolazione del colore dei fragment
2. interpolazione delle normali
3. raggio di riflessione esatto

Gli shader 1 e 2 erano già implementati, per realizzare il 3 è stato sufficiente calcolare il contributo di riflessione esatto usando la funzione `reflect` e sostituirlo all'approssimazione dell'half-way vector.

Si nota che le versioni 2 e 3 non sono molto diverse poichè l'half-way vector è una buona approssimazione del raggio di riflessione, tuttavia in alcune particolari posizioni è evidente la maggiore precisione del terzo metodo

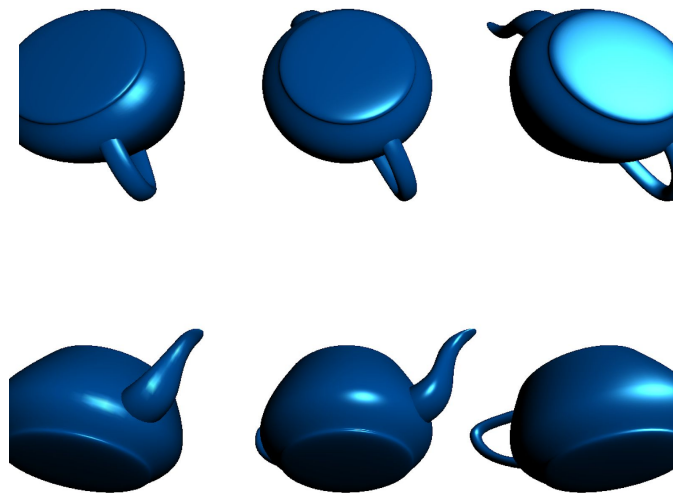


Figura 3. Confronto ordinato fra i 3 tipi di implementazioni di Phong

#### 4. Toon Shading

In questa esercitazione si analizza una tipologia di rendering non foto-realistico, ottenuto assegnando il colore di ogni pixel in base al dot product fra la normale e la direzione da cui proviene la luce.

È stata aggiunto un outline rosso ai pixel la cui normale forma un angolo circa 90 gradi con la camera, ossia quando il dot product fra la normale e il vettore direzione della camera è  $\leq 0$ .

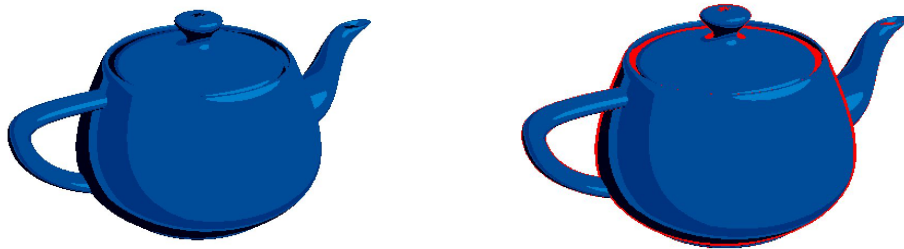


Figura 4. Rendering non fotorealistico senza bordo rosso (a) e con bordo rosso (b)

## 5. Morphing

L'esercitazione prevede l'utilizzo di un vertex shader per fare il morphing del colore e della forma di una figura.

È stata creata una animazione che da un triangolo con i vertici azzurro, rosso, blu fa il morphing ottenendo un quadrato con i vertici verde, giallo, rosso e verde scuro.

Per ottenerla è stata cambiata la primitiva da disegnare in `GL_POLYGON` ed il 4 vertice del quadrato è stato aggiunto in modo da farlo coincidere con il vertice in alto del triangolo.

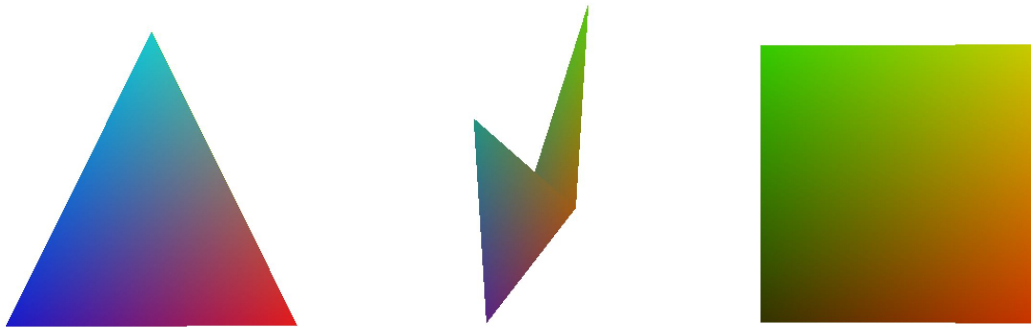


Figura 5. Triangolo iniziale (a), figura durante il morphing (b), quadrato finale (c)

## 6. Bump Mapping

L'esercitazione prevede l'applicazione di un bump texture procedurale su una mesh quadrata.

La texture generata disegna una griglia rialzata nel quadrato centrale della mesh e fa un solco lungo la diagonale  $i == j$

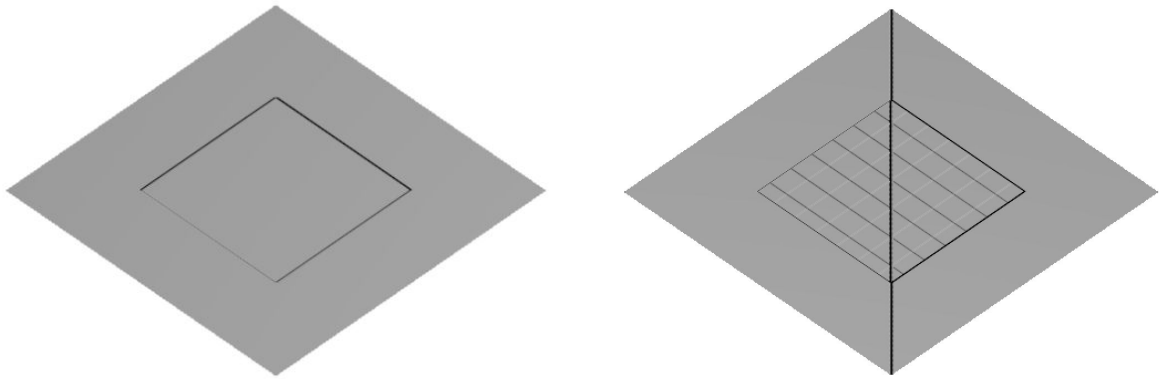


Figura 6. Bump mapping iniziale (a) risultato ottenuto (b)

## 7. Cube Environment Mapping

L'esercitazione aveva come scopo testare l'utilizzo environment mapping cubico. Trattandosi di una semplice applicazione di texture è bastato caricare su ogni lato del cubo una texture del planisfero.



Figura 7. Rendering della teapot che riflette le texture applicate all'environment cubico