Grafica al calcolatore Laboratorio – 5 effetti

Andrea Giachetti andrea.giachetti@univr.it
Marco Fattorel, Fabio Marco Caputo
Department of Computer Science, University of Verona, Italy

Programma esercitazioni

- Oggi: un po' di esercizi con shader, effetti
- Prossima volta: altri effetti, gestione collisioni
- Prossima volta: assegnamento seconda prova valutata
  - Partendo dagli esercizi base, alcune variazioni da effettuare, personalizzate
- Ultima lezione (31/5) Discussione degli esercizi con domande

Soluzione esercizi

- Eseguire e06
- Modificando i parametri nel file mtl per entrambi i modelli
  - Colorare l'oggetto di colore tendente al rosso variando le componenti diffusive
  - Attivare una luce ambientale di colore tendente all'azzurro. Cosa cambia?
  - Attivare la componente speculare. Cosa succede al variare di Ns? Cosa accade quando diventa alto?
- Eseguire e06b. Le due sfere si vedono in modo diverso? Come?
- Utilizzare la texture e modularla con la componente diffusiva e ambientale (non con la speculare)
  - Come cambierebbe modulando anche la speculare? provare

Phong shading

- Soluzione: passare le variabili al fragment shader e calcolare lì
  - Mandiamo in output posizioni e normali al fragment shader
    - "out vec3 Position;"
    - "out vec3 Normal;"
    - "out vec2 Coord;"
  - Carichiamo le altre variabili nel fragment
    - uniform float shininess;"
    - "uniform vec3 material_ambient;"
    - "uniform vec3 material_diffuse;"
    - "uniform vec3 material_specular;"
    - "uniform vec3 light_direction;"
    - "uniform vec3 view_position;"
  - Calcoliamo lì allo stesso modo (eventualmente poi facendo blending)

Effetto nebbia/distanza

- Programmando gli shader è relativamente facile ottenere alcuni degli effetti di cui abbiamo parlato a lezione
- Vediamo per esempio l'effetto nebbia o di sfumatura con lo sfondo a distanza
- Possiamo calcolarlo nel fragment shader, mescolando il risultato ottenuto prima con il colore di sfondo in funzione della distanza del frammento
- Possiamo ottenerlo da openGL. Codice e07.cpp
  - "float dist = gl_FragCoord.z / gl_FragCoord.w;"
    - Coordinata z nel SR camera
  - "float fogFactor = (25 – dist)/(25-0.01);" // si può fare di meglio...
  - "fogFactor = clamp(fogFactor, 0.0, 1.0);"
  - "outColor = mix(fogColor, outColor, fogFactor);"

Built in variables in GLSL

- https://www.opengl.org/wiki/Built-in_Variable_(GLSL)
- gl_FragCoord contiene le coordinate del frammento in spazio viewport

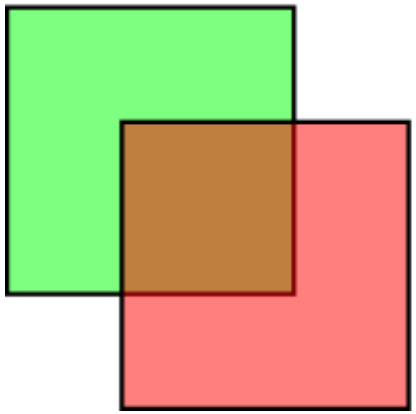
Trasparenza

- Abbiamo già codificato i colori usando il canale alfa
- Possiamo usare l'alfa-blending per ottenere l'effetto di trasparenza sugli oggetti. Ma attenzione: è molto problematico
- Vediamo senza preoccuparci troppo dei problemi...
- Occorre abilitare il blending col canale alfa (lo fa openGL)
  - glEnable(GL_BLEND);
- Definire la funzione con cui fa il blend
  - glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
- Non vogliamo che si faccia la rimozione con zbuffer
  - glDisable(GL_DEPTH_TEST);
- Potremmo voler vedere le facce posteriori!
  - glDisable(GL_CULL_FACE) ;

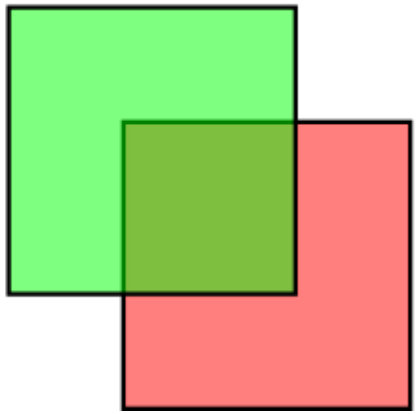
Limiti

- L'ordine è importante

Red on top

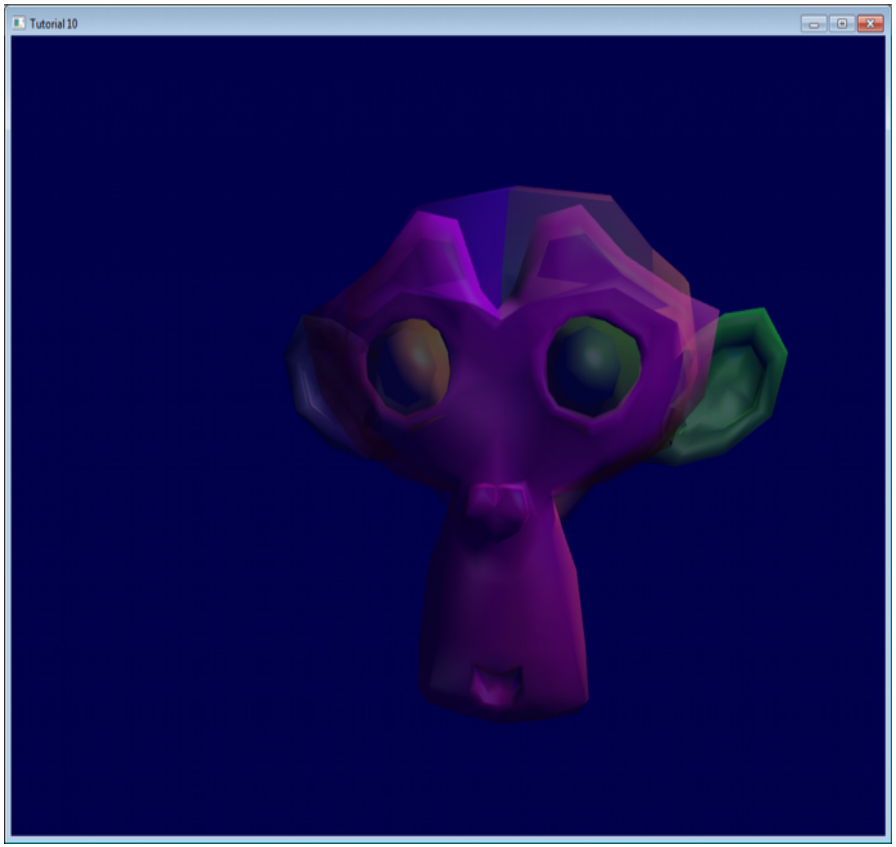


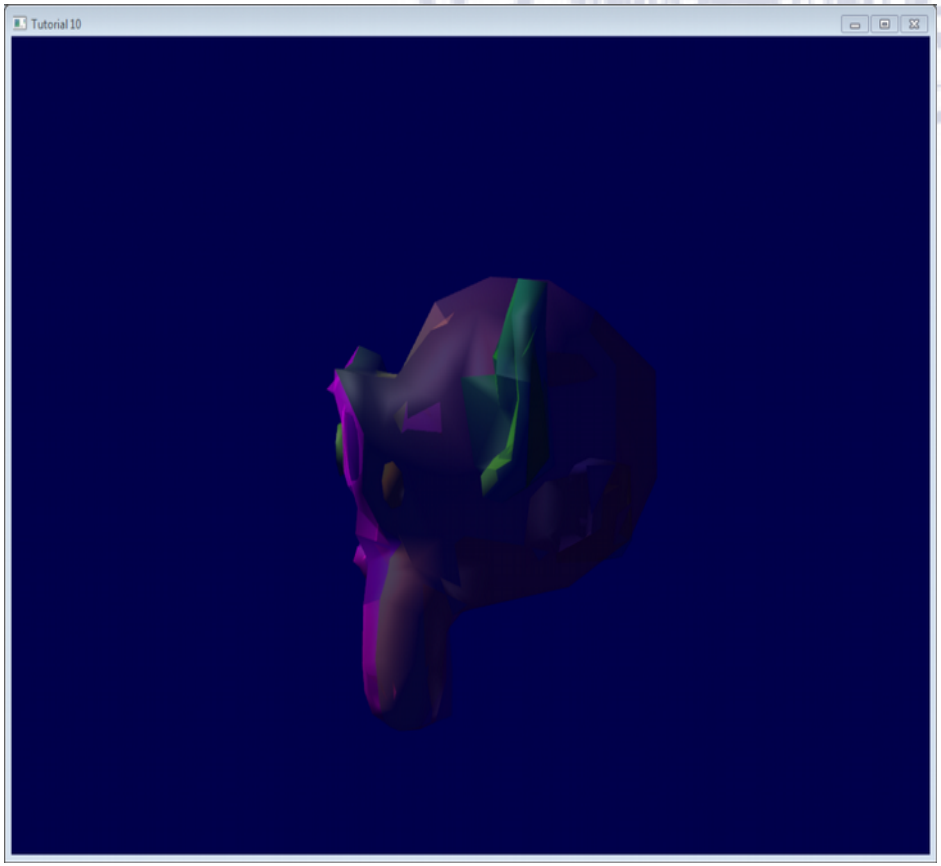
Green on top



Problemi

- Succedono cose di questo genere





Soluzione

- Difficile: occorrerebbe ordinare in profondità le primitive
  - Trremendamente complesso computazionalmente
- Limitare l'uso della trasparenza a situazioni in cui non ci sono effetti fastidiosi

Esercizio

- Partire dal file es08.cpp.
  - Togliere la trasparenza
  - Modificare la sfera con texture (terra), far orbitare attorno alla prima sul piano y=0, con orbita di raggio R=2
- Aggiungere una sorgente di luce puntiforme che illumina dalla posizione della sfera centrale. Attenzione
  - La luce che usavamo era direzionale: ora nello shader dovrò calcolare la direzione
  - Fare in modo che la terra abbia raggio 1/3 dell'originale
  - Occorre però fare attenzione...

Attenzione

- Se applico scalatura, la matrice che deve moltiplicare le normali non è in realtà la model matrix ma l'inversa della trasposta (si può dimostrare geometricamente)
  - normal_matrix = glm::transpose(glm::inverse(model)); // oppure normal_matrix = glm::inverseTranspose(model);
  - glUniformMatrix4fv(glGetUniformLocation(shaderProgram2,
  - "normal_matrix"), 1, GL_FALSE, &normal_matrix[0][0]);
    - Senza scalatura le due matrici coincidono
    - http://www.lighthouse3d.com/tutorials/glsl-tutorial/the-normal-matrix/

Esercizi

- Aggiungere una seconda sfera di raggio ½ che orbita con velocità doppia rispetto alla terra
  - Colore rosso
- Variare temporalmente il colore del sole tra giallo e rosso