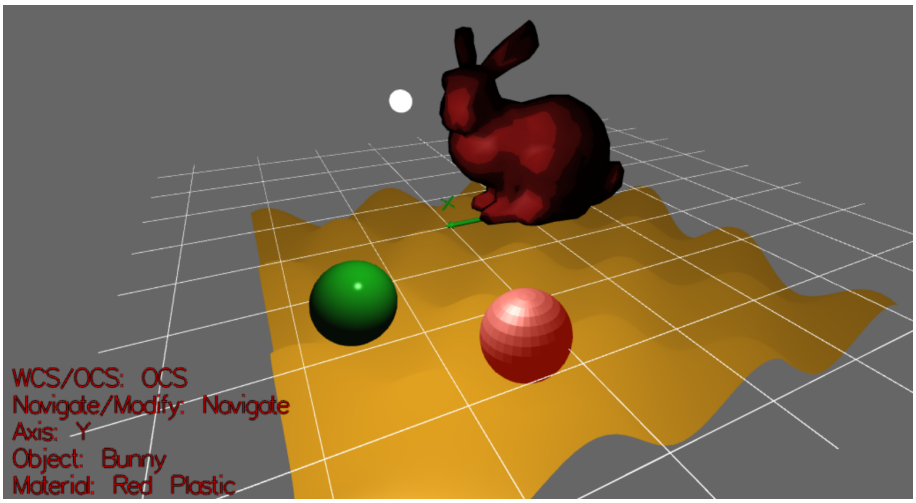
Computer Graphics - Relazione Laboratorio 3

Keran Jegasothy

Indice



1 Caricamento e visualizzazione modelli mesh poligonali

1.1 Nuovo materiale

É stato inserito un nuovo materiale con i paramenti riportati in seguito.

```
new_material_ambient = { 0.5, 0.05, 0.005 }, new_material_diffuse = { 0.5, 0.5, 0.5 },
new_material_specular = { 0.5, 0.5, 0.5}; GLfloat new_material_shininess = 5.5f;
```

1.2 Wave motion

Per realizzare il wave motion, sono stati realizzati:

1. un vertex shader (v_wave.glsl)
2. un fragment shader (f_wave.glsl)

Sia il vertex shader che il fragment shader sono stati realizzati modificando opportunamente il vertex shader e il fragment shader utilizzati per gouraud.

In particolare, è stato modificato la posizione della y dei vertici, applicando la seguente formula:

$$v_y = a \sin(\omega t + 10 v_x) \sin(\omega t + 10 v_z)\,,$$

1.3 Toon shading

Per realizzare il *Toon Shading*, sono stati generati due documenti:

1. un vertex shader
2. un fragment shader

Entrambi, sono stati realizzati partendo da del codice reperito in una repository ed opportunamente modificato.

2 Navigazione interattiva in scena

2.1 Movimento orizzontale della camera (sinistra/destra)

Per realizzare il movimento orizzontale della camera, è stato modificato opportunamente il codice che permette alla camera di effettuare il movimento verticale. In seguito è riportato il codice che permette alla camera di muoversi verso destra.

```
void moveCameraRight()
{
    glm::vec4 direction = ViewSetup.target - ViewSetup.position;
    glm::vec3 slide_vector = glm::normalize(glm::cross(glm::vec3(direction), glm::vec3(ViewSetup.upVector)));
    glm::vec3 upDirection = glm::normalize(slide_vector) * -CAMERA_TRASLATION_SPEED;
    ViewSetup.position += glm::vec4(upDirection, 0.0);
    ViewSetup.target += glm::vec4(upDirection, 0.0);
}
```

3 Trasformazione degli oggetti in scena

Le trasformazioni degli oggetti in scena vengono effettuate all'interno del metodo modifyModelMatrix. In particolare, all'interno di tale funzione vengono costruite le matrici di transazione, rotazione e scala.
A seconda del sistema di riferimento attivo (WCS o OCS), l'operazione varia.

```
switch (TransformMode) {
case WCS: objects[selected_obj].M = translation * rotation * scale * objects[selected_obj].M;
    break;
case OCS: objects[selected_obj].M = objects[selected_obj].M * scale * rotation * translation;
    break;
}
```