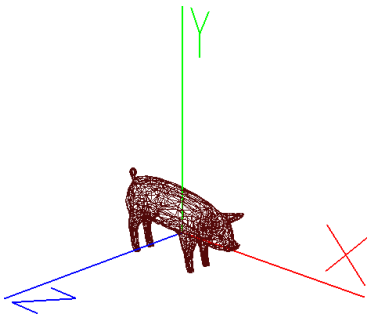
24 mar 2017

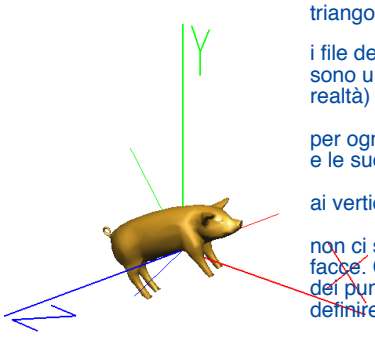
Fondamenti di Computer Graphics LM

Lab 2 - Navigazione interattiva in scena con modelli geometrici 3D

Questa esercitazione può essere eseguita sia in ambiente Windows che Linux. Dopo aver scaricato i file necessari, compilare ed eseguire il programma fornito.



a)



b)

le mesh che vediamo sono a triangoli

i file delle mesh sono in formato .m e sono una lista di vertex. (è un .obj in realtà)

per ogni vertice c'è l'indice del vertice e le sue coordinate.

ai vertici seguono le facce.

non ci sono gli edge, ma diamo le facce. Ci sono gli indici e hanno solo dei puntatori agli indici dei vertici per definire una faccia.

Se anche spostiamo l'oggetto, le facce rimangono le stesse (tranne se ci faccio un buco)

Figure 1: a) Uno screenshot dell'applicazione model_viewer; b) il rendering in model_viewer del modello pig.m in modalitá smooth dopo aver sviluppato il calcolo delle normali ai vertici, attivato lo shading smooth, cambiato il materiale e ruotato il modello.

1. **model_viewer - Caricamento e visualizzazione modelli geometrici**

   L'applicazione model_viewer carica e visualizza sia quadriche predefinite in OpenGL GLU library, sia oggetti mesh poligonali memorizzati in file con estensione .m. La directory `data` contiene vari file di modelli a mesh poligonali.

   Il programma model_viewer.c fornito presenta un menu pop up (a tendina) che si apre al click del bottone destro del mouse sulla finestra grafica e fornisce un ambiente interattivo di caricamento e visualizzazione di modelli geometrici 3D, sia primitive GLU che mesh poligonali in formato .m. Estendere il programma secondo le seguenti specifiche:

   (a) Caricamento e visualizzazione modelli geometrici di tipo mesh in formato .m

   (b) Visualizzazione superfici quadriche dalla libreria GLU (es. Sfere, cilindri, tori) non farlo, guardalo e basta

   (c) Verifica della gestione della visualizzazione dei modelli poligonali a mesh tramite display list.

   (d) Calcolo e memorizzazione delle normali ai vertici per i modelli mesh poligonali. Visualizzazione con normali ai vertici in modalitá smooth (`glShadeModel(GL_SMOOTH)`).
   basta fare solo le normali

2. **model_viewer - Controllo interattivo della scena**

   Alla pressione del tasto destro del mouse compare un menu pop up. Si può interagire con la scena selezionando gli item del menu e successivamente, tramite input da tastiera, modificare i valori associati. Le funzionalità che il sistema interattivo deve fornire sono elencate nella tabella 1, alcune

28 mar 2017

a = asse di rotazione (raggio) che dati due punti V e W è dato da:
VxW/||VxW|| intesi come distanze dal centro (crossprod)

ß = angolo di rotazione è acos(V*W)/(||V||*||W||)

VW = 2Rsina

a = ß/2

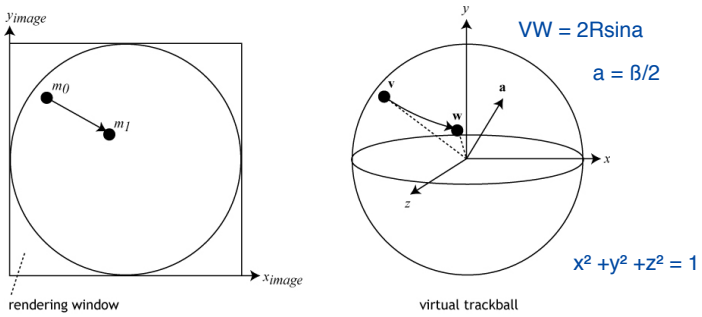


Figure 2: La trackball virtuale traduce il movimento del mouse in una matrice di rotazione

sono già realizzate (prendete visione di come) e altre invece sono lasciate da realizzare (indicate in grassetto). La trackball è già implementata, bisogna solo renderla SEMPRE attiva. quando mollo il click, ritorna all'inizio.

Aggiungere, ove necessario, le voci nel menu pop up. *Nota: A lettere piccole corrisponde un aumento del valore parametro, a lettere grandi una diminuzione.*

3. **Manipolazione dello stack delle matrici di trasformazione**

   Modificare l'applicazione model_viewer sviluppando i seguenti punti:

   (a) posizionare almeno tre diversi modelli geometrici nella scena sfruttando le funzioni openGL: `glPushMatrix(), glPopMatrix()`. ciascun elemento dovrà avere il suo push and Pop.

   (b) permettere la traslazione e rotazione dei **singoli** oggetti rispetto ai sistemi di riferimento WCS, OCS (World e Object Coordinate Systems). I singoli oggetti devono essere selezionabili tramite i tasti 1, 2, 3, etc. . Le trasformazioni di traslazione o rotazione devono essere applicate tramite i tasti 'x', 'X', 'y', 'Y', 'z', 'Z'. La selezione del tipo di trasformazione (traslazione / rotazione) e del sistema di riferimento rispetto al quale eseguire tale trasformazione deve essere selezionabile tramite i tasti:

questi tasti modificano posizione camE

   - 'o' traslazione rispetto all'OCS
   - 'O' rotazione rispetto all'OCS
   - 'w' traslazione rispetto al WCS
   - 'W' rotazione rispetto al WCS

premi il tasto e vai in uno stato e poi con frecce, muovi l'oggetto

4. ***OPZIONALE*: Manipolazione dello stack delle matrici**

   Modificare l'applicazione model_viewer per permettere la traslazione e rotazione dei **singoli** oggetti rispetto al sistema di riferimento VCS (View Coordinate System) seguendo le modalitá del punto 3. Il tipo di trasformazione (traslazione / rotazione) deve essere selezionabile tramite i tasti:

   - 'v' traslazione rispetto al VCS
   - 'V' rotazione rispetto al VCS (con assi di rotazione aventi direzione del VCS e passanti per l'origine del VCS oppure direzione del VCS e passanti per l'origine dell'OCS)

[2]La trackball virtuale permette di ruotare un oggetto (o meglio girare attorno ad un oggetto) interattivamente utilizzando il mouse. La figura 2 illustra intuitivamente come tradurre il movimento del mouse in un asse ed un angolo di rotazione. m_0 ed m_1 sono due posizioni consecutive del mouse e definiscono due punti v e w sulla semisfera 3D virtuale. Il loro prodotto vettoriale definisce l'asse di rotazione $a = v \times w$, mentre l'angolo di rotazione può essere calcolato dal loro prodotto scalare.

[2]Muovere la camera virtuale lungo un percorso in modalità *look at* (mediante curva chiusa di Bézier) in modo che la camera si muova lungo un percorso con centro di interesse l'oggetto in scena

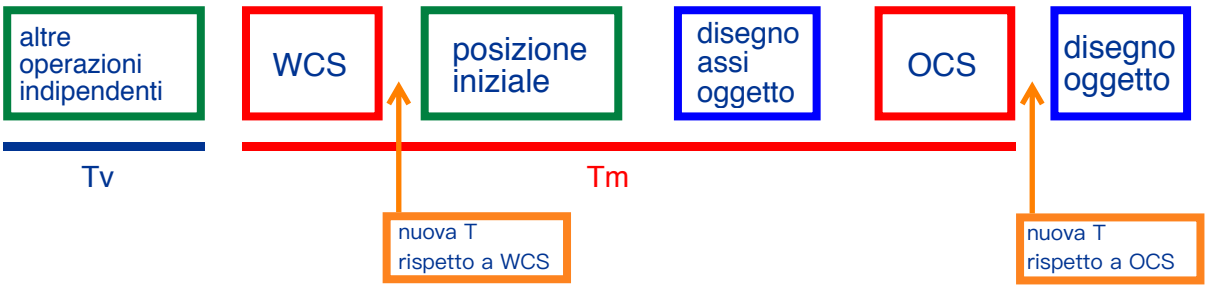
per le traslazioni dobbiamo tenere una matrice OCS[#mesh][16]. Se io ho una trasformazione T, e faccio un'operazione: T × OCS → ho come risultato, un'operazione a livello di mondo. Quindi:

W: OCS = T*OCS (disegno oggetto) → trasformazione mondo
O: OCS = OCS*T (disegno oggetto) → trasformazione oggetto

2

praticamente:
OCS:
glPushMatrix()
glLoadIdentity()
glMultMatrix(OCS[obj])
glTranslate(trans[0],trans[1],trans[2])
T gRotate(rot[0],1,0,0);
gRotate(rot[1],0,1,0);
gRotate(rot[2],0,0,1);
glGetFloatv(OCS[obj]);
//metto dentro la OCS il risultato di T*OCS

questo è una trasformazione a livello di oggetto, OCS.
se voglio la trasformazione a livello di mondo, WCS, inverto le 4 funzioni della T e glMulMatrix.



Funzionalità	*Descrizione*	*Tasti*	*Default*
Change eye point	modifica le coordinate del punto di vista	x,X,y,Y,z,Z	(8.8, 4.9, 9.0)
Change ref. point	modifica le coordinate del punto di riferimento del sistema	x,X,y,Y,z,Z	(0.0, 0.0, 0.0)
Change up vector	modifica le coordinate del vettore "alto" della camera	x,X,y,Y,z,Z	(0.0, 1.0, 0.0)
Change light position	modifica le coordinate del vettore posizione della fonte luminosa	x,X,y,Y,z,Z	(5.0, 5.0, 5.0)
Rotate model	ruota il modello	x,X,y,Y,z,Z	(0.0, 0.0, 0.0)
Zoom in/out	modifica l'angolo (field of view) al vertice della piramide di vista	f,F	20.00
Proiezione	modifica il tipo di proiezione e quindi il volume di vista	-	prospettica
Culling	abilita/ disabilita il back face culling	vedo o non vedo i triangoli che sono dietro	off
Wireframe	cambia modalita' di rendering dei poligoni	-	GL_LINE
Shading	Cambia modalita' di shading	-	GL_FLAT
Materials	Cambia il materiale associato agli oggetti	-	red_plastic
Trackball	modalità trackball virtuale per il controllo della camera. Vedi nota[1]	-	-
Camera Motion	movimento della camera lungo un percorso. Vedi nota[2]	-	-
Print system status	stampa sul terminale i dati correnti del sistema	-	-
Reset	riporta il sistema allo stato iniziale	-	-
Quit	chiude il programma	Esc	-

il punto di vista deve essere cambiato. camE

si crea una curva di bezier chiusa (non interattiva) per spostare la camera sulla curva (che non si deve vedere)

probabilmente si deve spostare la camera con mouse o frecce

Table 1: Tabella delle funzionalità sviluppate e da sviluppare (indicate in grassetto) nell'applicazione model_viewer