INFORMATICA GRAFICA E MULTIMEDIALITA'

PROGETTO 3 - A.A. 2008/2009

"Fuga verso l'Ultimo Avamposto"

Livello: Difficile

SPECIFICHE:
L'applicazione deve visualizzare, in vista soggettiva, un paesaggio tridimensionale, opportunamente illuminato da una sorgente luminosa ("sole"), sul quale ci si possa muovere premendo opportuni tasti sulla tastiera (si implementi almeno la possibilita' di muoversi avanti, indietro e di ruotare sull'asse Y a destra o sinistra).
Il paesaggio sara' rappresentato semplicemente da una superficie bidimensionale di colore verde (si posizioni pero' il "sole" con un angolo opportuno rispetto alla verticale in modo da produrre una sufficiente ombreggiatura sul terreno, onde poterne distinguere le asperita').
Tale superficie viene creata caricando un'immagine bidimensionale in bianco e nero (1024x1024x1 byte) che rappresenta punto per punto (con risoluzione di un metro per pixel) l'elevazione del terreno, secondo la codifica che 0 (nero) rappresenta un punto a quota 0 metri sul livello del mare e 255 (bianco) rappresenta un punto a quota 25.5 metri. Tale immagine deve essere contenuta nel file binario "terreno.map" presente nella stessa directory del programma, e deve essere opportunamente creata; si lascia completa liberta' sul come realizzare tale file, se generarlo con algoritmi pseudo-casuali o utilizzare programmi di disegno, purche' la mappa finale sia sufficientemente movimentata nell'elevazione (non sono accettabili terreni completamente "piatti").
In un punto del terreno deve essere collocato un oggetto (anche un semplice quadrato orizzontale di dimensioni opportune), di colore diverso e ben distinguibile dal terreno, rappresentante la zona da raggiungere; tale oggetto deve, ovviamente, essere posto in una posizione diversa rispetto alle coordinate di partenza dell'osservatore, possibilmente in maniera tale da renderlo nascosto alla vista a causa del terreno accidentato.
Scopo dell'applicazione e' trovare e raggiungere, tramite i tasti di controllo dell'osservatore, il suddetto oggetto. Il compito deve pero' essere complicato dalla presenza di un certo numero di "nemici", oggetti mobili (che possono venir rappresentati anche da semplici primitive sferiche) collocati sul terreno (a coordinate sufficientemente distanti, in partenza, dall'osservatore) il cui scopo e' quello di raggiungere l'osservatore prima che questi riesca ad arrivare al punto prefissato.
Si lascia completa liberta' di decidere la "strategia" in base alla quale i "nemici" cercano di raggiungere l'osservatore, purche' naturalmente vi sia alla base una logica ragionevole (non e', ad esempio, una buona soluzione il puro movimento casuale). Si presti attenzione al fatto che la velocita' di spostamento dei "nemici" dovra' essere sufficiente per rendere interessante la fuga, ma non eccessiva.

FACOLTATIVAMENTE:
1) si vincoli la velocita' di movimento sia dei nemici che dell'osservatore al

dislivello affrontato localmente - in pianura la velocita' deve essere maggiore rispetto alla salita, ed oltre una certa pendenza (45 gradi, ad esempio) non dovrebbe essere proprio possibile arrampicarsi
2) si implementi una strategia dei "nemici" che tenga conto della "linea di vista", ovvero del fatto che l'osservatore sia nella loro visuale o meno (perche' nascosto dalle asperita' del paesaggio); ad esempio, i "nemici" inseguono l'osservatore solo se lo vedono, altrimenti si aggirano casualmente per il territorio
3) si utilizzi la tecnica del texture-mapping per migliorare la resa grafica del paesaggio e/o degli altri oggetti
4) si implementi la possibilita' di dotare sia l'osservatore che i nemici di "fucile laser" in grado di colpire l'avversario; il "fucile laser" puo' colpire solo se la "linea di fuoco" e' libera (non ci devono essere oggetti ne' asperita' del paesaggio tra il fucile ed il bersaglio); per colpire e' necessario mirare il bersaglio (si implementino gli opportuni movimenti necessari all'osservatore); per i "nemici" si studi un algoritmo di "errore" che aumenti la probabilita' di "smirare" in funzione della distanza del bersaglio; si visualizzi, naturalmente, il "raggio" del laser quando il fucile viene azionato (sia da parte dell'osservatore che da parte dei nemici).

REALIZZAZIONE:
Per la realizzazione si faccia uso della glut.library o di librerie di accesso all'OpenGL equivalenti.
L'applicazione rappresenta il classico problema che puo' ben essere scomposto nel paradigma MVC (Modello - Visualizzazione - Controllo); in altre parole, vi dovranno essere tre parti ben distinte nel programma: la parte che si occupera' del Modello (movimento osservatore/"nemici" e controllo collisioni), la parte che si occupera' della Visualizzazione della scena e la parte che gestira' il Controllo dell'osservatore tramite l'input da tastiera dell'utente.
Utilizzando la glut, una scelta "naturale" potrebbe essere quella di gestire il Modello nella funzione di call-back passata alla glutIdleFunc(), la Visualizzazione nella call-back della glutDisplayFunc() ed il Controllo con la glutKeyboardFunc().

Nella parte principale dell'applicazione (main) si dovrà configurare un contesto OpenGL (inizializzazione/apertura della finestra, etc.) e caricare il file di descrizione del paesaggio, costruendo opportune strutture dati per mantenere le informazioni così lette (si consiglia, a questo proposito, di creare una matrice 1024x1024 atta a contenere punto per punto l'elevazione del terreno). Si dovranno poi settare le funzioni di call-back per il ridimensionamento della finestra ed il tracciamento della grafica nella finestra stessa (si faccia riferimento alla documentazione della libreria che si intende usare).

La funzione call-back passata alla glutReshapeFunc (o chi per essa) dovrà configurare la matrice di proiezione per ottenere una opportuna distorsione prospettica (si può utilizzare a tale proposito la gluPerspective), l'angolo focale che si vuole utilizzare per la telecamera è libero (si suggerisce un angolo di ~45 gradi).

La funzione call-back passata alla glutIdleFunc (o chi per essa) dovra' aggiornare la posizione dell'osservatore e dei "nemici", controllando le eventuali collisioni. Un metodo possibile puo' essere quello di utilizzare 4 variabili di stato

cinematiche per ciascun oggetto mobile, due per la posizione sul piano X, Z e due per le velocita' lungo gli assi coordinati VX e VZ (la posizione su Y e' univocamente determinata dalla superficie del terreno, dal momento che nessun oggetto puo' "volare").
L'aggiornamento della posizione puo' essere fatto con la semplice "formula di Eulero", ovvero:

X = X + (VX * DT);
Z = Z + (VZ * DT);

che non rappresenta altro che il calcolo del moto rettilineo uniforme nel tempo DT. Tale intervallo di tempo e', ovviamente, il tempo trascorso tra la chiamata precedente alla call-back e la chiamata attuale, e puo' essere determinato utilizzando lo statement glutGet(GLUT_ELAPSED_TIME), che ritorna il tempo trascorso (in millisecondi) dalla partenza dell'applicazione; salvando tale valore in una opportuna variabile globale e confrontandolo con il valore trovato nella chiamata precedente e' possibile determinare il numero di millisecondi trascorsi (cioe, il DT) tra una chiamata e l'altra alla funzione di call-back del Modello.
Per determinare le velocita' VX e VZ dei nemici si dovra' determinare una strategia atta a raggiungere l'osservatore (ad esempio utilizzando il vettore distanza).
Il controllo delle collisioni potra' essere fatto semplicemente valutando le distanze tra le coordinate X, Z dei vari oggetti.

La funzione di call-back associata alla tastiera dovra' tener conto dello spostamento dell'osservatore, modificando quindi SOLO il suo vettore velocita' VX, VZ e l'angolo di rotazione su Y.

La funzione di call-back passata alla glutDisplayFunc dovra' infine visualizzare il Modello dal punto di vista dell'osservatore: in particolare si dovra' spostare l'osservatore nel punto corretto (utilizzando le coordinate X, Z e determinando Y dalla mappa del terreno), attivare la sorgente luminosa atta a simulare il sole, attivare il materiale del terreno e visualizzare la superficie del terreno scegliendo un opportuno metodo, nonche' visualizzare i "nemici" e/o altri oggetti presenti sulla scena utilizzando i relativi materiali/texture (anche per i "nemici" il posizionamento dovra' avvenire grazie alle coordinate X, Z determinando Y dall'elevazione del terreno).