Interactive 3D Graphics

Palla Federico, Zilli Francesco

June 2016

Introduzione

Lo scopo principale del nostro progetto è quello di generare una variante 3D del `flash-game "Bowman"`. Nella versione proposta, il protagonista, al fine di non essere ucciso, deve colpire più volte con il suo arco un mostro che si dirige verso di lui.

All'avvio della pagina web, viene quindi mostrata la scena principale all'utente, utilizzando una camera mobile che termina il suo movimento una volta che il mostro entra nel campo visivo (in questa fase è possibile impostare la distanza del target che inizierà subito ad avvicinarsi). Una volta terminato il video introduttivo, viene attivata la camera principale che inquadra l'arco del protagonista, configurabile mediante movimento del mouse e pannello `datGUI` (su questa è anche presente la distanza aggiornata del mostro).
Scelto l'allungo ed il libraggio dell'arma, alla pressione del tasto `space` viene tolto il blocco della freccia, ed al successivo `click sinistro` del mouse verrà effettuato il lancio vero e proprio.

Nell'inquadratura principale (mainCamera) troviamo anche:

- un carro di legno, sul quale sono posizionate le frecce rimanenti;
- un lampione, che a seconda del colore nella lanterna indica la presenza o meno del blocco relativo alla freccia;
- tre colonne, sulle quali verrà posizionata una statuetta una volta colpito il mostro;

Shaders

Per applicare le texture agli oggetti presenti nella scena, abbiamo implementato tre shaders, che verranno in seguito descritti. Ciascuno shader tiene in considerazione le quattro luci implementate.

Lambertian

Questo tipo di shader (situato in */assets/shaders/lambertianShader.js*) implementa uno shader Lambertiano con l'aggiunta della normal map, per tenere traccia delle perturbazioni delle normali nei punti sulla superficie.
Gran parte della computazione viene svolta nel fragment shader, in cui per generare il colore finale sommiamo la quantità di luce proveniente dall'ambiente al colore calcolato. Abbiamo scelto di implementare questo tipo di shader per migliorare le prestazioni complessive del gioco.

Microfacet

Questo shader (situato in */assets/shaders/microFacetShader.js*) implementa uno shader che si basa sull'utilizzo delle Micro-Facets per calcolare il termine speculare e, quindi, la riflessione della luce.
Come per il Lambertian, è stato aggiunto il calcolo delle normali memorizzate nella `normal map`, messe più in risalto grazie al termine speculare.
Le tre funzioni per il calcolo del BRDF (riflettanza di Fresnel con approssimazione di Schlick F, geometry term G e distribuzione delle normali D) sono state implementate come visto a lezione.

Ground

Nella generazione del terreno, per contenere eventuali cali di prestazione, viene applicato un `Lambert-shader` esclusivamente ad un insieme ristretto di texture. Su queste ultime viene applicata una displacement map, al fine di simulare la presenza di fili d'erba.
Allo stesso modo, in prossimità dell'arco, sono stati aggiunte delle texture che simulano la presenza di blocchi in cemento per migliorare l'ambiente circostante.

Luci

Vi sono tre tipi di luce nella scena:

- Point light: sono le luci usate per implementare le luci rosse e verdi del lampione. La formula della radianza è:

$$\beta = \frac{lightPower}{4 \cdot \pi \cdot length(lightVector)^2} \tag{1}$$

  dove lightPower è un valore fisso inviato al fragment shader come uniform, lightVector viene calcolato dal vertex shader.

- Directional light: implementate impostando il vettore della direzione ad un valore fisso. La formula della radianza è la stessa delle Point light.

- Spot light: abbiamo implementato una Spot Light in modo da poter evidenziare la mesh dell'arco e la postazione di tiro dell'arciere. La formula

per il calcolo della radianza è simile a quella delle Point Light solo che viene implementata come un sistema di tre equazioni, per poter simulare le condizioni di illuminazione fuori, dentro nell'anello più esterno e nell'anello più interno.

Postprocessing

Nel progetto viene impiegato un effetto di `desaturazione` e di `color mixing`. Se il mostro viene colpito, il colore della scena viene prima desaturato, poi interpolato ad un colore rosso, al fine di sottolineare la morte della creatura.

Animazioni

Il mostro dispone di tre animazioni: camminata, attacco e morte. Queste sono state generate tramite key-framing dell'armatura associata in `Blender`, ed esportate in formato `dae`; vengono inserite nella scena al caricamento della pagina web e rese visibili in base alla logica booleana del gioco.

Ogni volta che viene colpito il mostro, viene animato un testo (texture su oggetto trasparente) e caricata una statuetta, la cui rotazione è ricavata dal collegamento di un quaternion con la mesh.

Modelli

I modelli .obj presenti nel gioco sono stati creati (colonne, lampione) o scaricati dal sito `tf3dm` (carretto, arco, freccia). Il modello del mostro, invece, è stato scaricato dagli esempi online di `three.js` ed animato con Blender.
Tutte le texture sono state reperite online, ed eventuali normal, specular e displacement map generate usando il sito `NormalMapOnline`

Suoni

Nel gioco sono stati inseriti anche tre effetti sonori, generati dalla combinazione di più suoni in Blender (modalità Video Sequencer):

- `monster_dies.wav`, riprodotto durante la killCam, alla morte del mostro;
- `archer_scream.wav`, lanciato una volta che l'arciere viene raggiunto dal mostro;
- `fail_sound.wav`, udibile quando la freccia colpisce il terreno o esce dal campo di gioco;