Relazione sul progetto per il corso di Interactive 3D Graphics

Università degli studi di Udine

Autori: Rostyslav Kostyuk, Riccardo Lulli
Nome progetto: OLA (Oh, Look an Asteroid)
Link GitHub del progetto: https://github.com/DudduYum/RS

Introduzione
Il progetto OLA è un progetto di grafica 3D interattiva, nello specifico una demo tecnica (ma giocabile) di un videogioco, eseguito tramite WebGL e codificato usando le librerie di three.js.

Meccaniche di gioco
Si tratta di un endless game (non termina mai salvo game over) nel quale un giocatore deve muovere l'astronave per evitare gli asteroidi in avvicinamento.

Comandi
- [barra spaziatrice] : avvia una nuova partita (sia da schermata di iniziale che da game over) oppure immobilizza l'astronave durante il gioco
- W,A,S,D e frecce direzionali : muovi l'astronave
- P : pausa
- C : passa da una telecamera all'altra (di gioco o libera)

Implementazioni grafiche
Texture
Per migliorare l'aspetto degli oggetti presenti, sono state utilizzate delle texture per
- colorazione
- *displacement mapping*
- *normal mapping*
- *specular mapping* (rendere alcuni punti della superficie più riflettenti rispetto alle altri)
- *texture animation*

Per creare la normal, *displacement* e *specular map* abbiamo utilizzato il seguente sito
https://cpetry.github.io/NormalMap-Online/
che permette di creare le mappe di distorsione della texture desiderata, ottenendo così delle perturbazioni della superficie coerenti con la texture.
Su ciascuna texture abbiamo attivato un filtro bilineare per risolvere la *magnification*, un filtro trilineare per risolvere la *minification* e impostato il livello massimo del filtro anisotropico (valore max dato dalla GPU).

Luci
Abbiamo implementato tre luci di due tipi nel nostro progetto:
- *point light*, utilizzata per modellare il sole e la fiamma della navicella
- *ambient light*, per correggere la hard shadow create dal sole e fiamma

Come modello di luce abbiamo implementato la Lambertian BRDF vista a lezione, utilizzando però più sorgenti di luce.

Shader

Tutti gli shader utilizzati nel progetto sono implementati tramite shaderMaterial fornito da THREE.js. Li shader principali sono:

- *asteroid shader* : implementano tutte le tecniche descritti nella sezione “Texture”
- *spaceship shader :* implementano texturing ed eseguono la colorazione del materiale in base al colore impostato
- *flame shader* : implementa *displacement mapping* e *texture animation* di cui parleremo nella sezione “Animazione”

Post-processing

Gli effetti di post-processing sono applicati usando il fragment shader. Per concatenare l’applicazione di shader è stata usata la libreria EffectComposer (in congiunzione a RenderPass, ShaderPass e CopyShader). Questa libreria permettere di aggiungere vari passaggi, ciascuno dei quali ottiene in input una texture 2D che è il risultato del passaggio precedente.
Il primo passaggio è il rendering della scena stessa. L’immagine ottenuta del render viene passata ai shader successivi.

- **Dynamic Depth of Field**
  L’effetto DOF è una sfocatura degli oggetti fuori fuoco, lontani dalla distanza focale della lente.
  Un DOF statico determina anticipatamente gli elementi da sfocare e tale effetto non cambia al variare della distanza (es. un paesaggio in lontananza è sempre sfocato alla stessa maniera). Un DOF dinamico invece considera ad ogni ciclo di rendering la distanza degli oggetti rispetto al punto di fuoco e li sfuoca di conseguenza.
  La distanza degli oggetti dalla telecamera è letta da una texture 2D prodotta dal rendering di un EffectComposer secondario, dove viene usato un materiale ad hoc per tutti gli oggetti presenti nella scena. Tale materiale colora un oggetto in base alla sua distanza dalla telecamera. Il risultato del rendering con tale materiale (applicato a tutti gli oggetti) è una mappa di profondità.
  Per ogni pixel viene presa in considerazione un’area 11x11 dalla quale leggere i colori dei pixel adiacenti. I colori vengono pesati tramite una matrice di convoluzione di Gauss con sigma pari a 2,0.
  Per migliore l’efficienza, il filtro di convoluzione è separato in due vettori e vengono effettuati due passaggi nel composer: il primo applica un filtro verticale, il secondo invece orizzontale.
  Il risultato è identico a quello ottenuto con una matrice 11x11, però i calcoli effettuati per pixel sono 22 invece di 121 (che sarebbe computazionalmente oneroso).
  In fine viene scelto un colore nel mezzo (interpolazione) tra quello originario del pixel e il colore di massima sfocatura: il grado di interpolazione dipende dalla distanza dell’oggetto rispetto al punto focale.

- **Pixelation**
  L’effetto di pixelation consiste nel rendere l’immagine “pixelosa”, come se fosse visualizzata ad una risoluzione inferiore. Il colore di un pixel è uguale alla media dei colori presenti nel “pixel grande”, ovvero un’area quadrata del quale il pixel fa parte (l’appartenenza ad un’area è determinata tramite operazione modulo sulle coordinate del pixel).
  Le dimensioni massime dei “pixel grandi” sono 8x8, con valori superiori il costo computazionale è oneroso per schede grafiche di fascia bassa.

- **Edge detection**
  Semplice implementazione del metodo di edge detection di Sobel.

- **Image settings**
  Questo shader dà la possibilità di manipolare luminosità e contrasto dell’immagine.

Animazione

Per rendere l'ambiente di gioco più gradevole da vedere abbiamo aggiunto alcune animazioni agli asteroidi e alla fiamma dell'astronave. Nella nuova release gli asteroidi si muovono lungo una traiettoria casuale scelta al momento di spawn. Per rendere gli asteroidi più verosimili abbiamo aggiunto una rotazione che varia in base alla loro dimensione, usando i quaternioni. L'animazione viene eseguita all'interno del metodo *move* della classe asteroide. Per mantenere l'asteroide sulla propria traiettoria, al suo vettore di spostamento viene applicato l'inverso del quaternione applicato alla mesh del asteroide.
L'animazione della fiamma è realizzata usando la sua *displacement map*. Sfruttando il fatto che la densità della *displacement map* è più alta rispetto all'oggetto in considerazione, incrementandone l'offset si può ottenere un interessante effetto che assomiglia al comportamento di una fiamma.

Per rendere la schermata iniziale più interessante abbiamo aggiunto un movimento della camera. Questa animazione viene generata all'avvio del gioco ed è pseudo casuale. Per realizzarla abbiamo introdotto un oggetto che prende in input 2 coordinate, centro e punto d'osservazione. Con questi valori viene generata una traiettoria "orbitale". In realtà si tratta di una circonferenza perturbata, i punti sono perturbati con delle funzioni pseudo casuali.

Tutte le animazioni sono basate sullo scorrere del tempo e non sul framerate. Questo garantisce la medesima esperienza visiva e ludica a prescindere dal monitor e dal hardware.

Altro

Le meccaniche di gioco sono tutte basate sullo scorrere del tempo e non sul framerate. Questo permette di giocare a qualsiasi framerate senza alterare l'esperienza di gioco o causare problemi di funzionamento.
Il gioco possiede una difficoltà crescente che aumenta ogni 3 secondi, raggiungendo l'apice a 60 secondi. Ad ogni incremento di difficoltà la velocità degli asteroidi aumenta e si riduce il tempo di spawn tra un asteroide e il successivo.
Il gioco può essere messo in pausa premendo il tasto P, in modo da poter osservare la scena liberamente.
Al gioco è stata aggiunta una canzone di background e il suono di un'esplosione nel caso di impatto con un asteroide. I suoni possono essere regolati tramite il pannello delle impostazioni.