INTRODUZIONE A AIV.FAST2D

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

AIV.DRAW BASSE PRESTAZIONI

La metodologia di disegno vista in Aiv.Draw è la metodologia “old school”.

Questa metodologia è poco prestante. Era utilizzata prima dell’invenzione delle schede video moderne.

Perché poco prestante? Qualcuno mi sa dire tutta la parte in cui veniva effettivamente calcolato quello che poi vedevamo, da quale componente era eseguito?

GRAFICA ACCELERATA

Dalla CPU. Tutti i calcoli venivano effettuati dalla CPU perché era codice C#.

L'acceleratore hardware (o acceleratore fisico), è un componente hardware per computer progettato per migliorare le prestazioni del PC in settori specifici.

La scheda video è un acceleratore hardware per operazioni di calcolo in virgola mobile o elaborazioni grafiche. Infatti qual è l'unità di misura con cui si misura la prestazione di una scheda video? (TIP: sta diventando molto di moda da quando le console nell'ultima mezza generazione e nella nuova generazione hanno iniziato a giocare "a chi spara più lontano").

PRIMA

Quello che abbiamo fatto fino ad ora con la GPU non è stato nient'altro che trasferire le informazioni dei pixel dalla memoria del nostro programma, alla memoria della GPU (blitting).

Questo trasferimento dati è molto lento e inoltre la computazione di tutti i dati da trasferire è a carico della CPU.

Le GPU moderne sono molto, molto, molto efficaci

GPU MODERNE

Le GPU moderne sono molto, molto, molto efficaci nell'eseguire alcuni tipi di calcoli, tra cui:

- Rasterizzazione: dato un triangolo (3 vertici nello spazio) riempire l'area del triangolo in un certo modo.
- Interpolazione lineare: dati due punti trovare i valori intermedi

Possiamo caricare al loro interno del codice, cioè delle istruzioni che saranno eseguite all'interno della GPU (non della CPU!). Esistono due tipi di programmi che possiamo caricare nella GPU:

- vertex shader: che vanno a modificare la posizione dei vertici.
- pixel shader: che definiscono il colore del pixel.

La comunicazione tra CPU e GPU avviene attraverso una DRAW CALL.

GPU MODERNE

Questi particolari programmi vengono eseguiti ogni frame.

In particolare, in OGNI frame, il vertex shader viene eseguito una volta PER OGNI vertice visibile nella camera in quel momento, e il pixel shader viene eseguito una volta PER OGNI PIXEL. (Se pensate alla risoluzione 4K, capite subito la quantità di volte che vengono eseguiti, in ogni frame).

AIV.FAST2D

Aiv.Fast2D è l'engine che utilizzeremo per il resto dell'anno (fatta eccezione per le ultime lezioni). Questo engine è un engine molto base (come Aiv.Draw) ma sfrutta l'accelerazione hardware delle GPU.

AIV.FAST2D - MESH

Alcuni concetti sono fondamentali:

- non dobbiamo solo chiedere al sistema operativo di accedere a una finestra, ma anche alla GPU.
- dobbiamo avere la possibilità di creare dati direttamente all'interno della GPU (per evitare il trasferimento di dati). Abbiamo però bisogno di avere un riferimento di questi dati. E questo riferimento è il loro "nome" (che in realtà è un numero). In Aiv.Fast2D riusciamo a fare questo attraverso la classe Mesh (che sarebbe un buffer di dati dalla GPU). Una mesh solitamente rappresenta un insieme di triangoli (appunto una "maglia).

Istanziando una mesh ho quindi un nome ma non ancora dei dati. Ho solo creato un riferimento a dei possibili dati nella GPU.

AIV.FAST2D - MESH

All’interno della mesh abbiamo diversi attributi che possiamo utilizzare:

- v: che sta per vertice. Questo è un array di float che rappresenta la posizione di tutti i vertici della mesh.
- uv: (vediamo tra due slide).

AIV.FAST2D - DRAW CALL

La Draw Call viene fatta in questi step:

- vertex shader: prende le coordinate dei vertici (che abbiamo indicato in pixel) e le trasforma nel dominio -1,1 (dominio con cui lavora la GPU). L'output di questo step sono le coordinate del vertice. (Se sono fuori dal dominio -1,1 non si vedranno perché sono fuori dalla finestra).
- rasterizzazione: interpolazione lineare tra due punti per trovare il colore di tutti i pixel
- pixel shader (o fragment shader): programma che viene eseguito per ogni pixel

FRONT BUFFER E BACK BUFFER

Man mano che vengono eseguiti i vari shader, si andrà a disegnare l'output di questi shader in un buffer (che possiamo vedere come la nostra bitmap in Aiv.Draw).

C'è però un problema, se la durata del frame è maggiore ad esempio della durata del refresh rate del monitor, con una soluzione a un solo buffer, vedremmo il frame che si disegna poco alla volta.

Per risolvere questa cosa, esistono due buffer, che funzionano in questo modo:

- Front buffer: è il buffer che rappresenta quello che viene mostrato in output sul monitor, e cioè le informazioni dell'ultimo frame ad essere stato renderizzato.
- Back buffer: è il buffer in cui viene disegnato man mano il nuovo frame.

Ad un certo punto, quando il nuovo frame sarà pronto, i due buffer si invertiranno, quindi il back buffer diventerà il front buffer e front buffer diventerà il back buffer. E così via per ogni frame...

ESERCIZIO 1

Disegniamo un quadrato rosso!

AIV.FAST2D - TEXTURE

Direi che siamo stufi di disegnare forme con colori pieni no?

Chiariamo cosa sono le UV.

Le UV sono delle coordinate che corrispondono a dei punti di ancoraggio di un'immagine. Si chiamano coordinate UV perché X e Y erano già prese :)

Una Texture è una classe che permette di caricare (direttamente nella GPU) un'immagine, e avere una reference a questa immagine.

Con le coordinate UV mappiamo ogni mesh con una determinata porzione di quella immagine.

ESERCIZIO 2

Disegniamo un rettangolo con una texture! Proviamo prima a disegnare la texture sullo stesso rettangolo dell'esercizio 1 e poi proviamo a capire come creare un rettangolo delle dimensioni adatte a non streatchare la texture.

ESERCIZIO 2 BIS

Proviamo a disegnare solamente la prima parte della texture sul rettangolo 100x100 per evitare che ci sia la intera texture streacthata.

AIV.FAST2D - SPRITE

Ma se noi vogliamo disegnare sempre oggetti 2D (quindi due triangoli con una Texture), ogni volta devo fare tutto questo processo arzigogolato?

La prima (e credo unica) astrazione in Aiv.Fast2D, che ci semplifica un po' la vita, è il concetto di Sprite → Rettangolo con UV mappate sulla totalità della texture.

Attenzione che qui (a differenza di Aiv.Draw e Unity3D) per Sprite si intende una forma rettangolare a cui è associato un nome nella GPU. Non alla bitmap di una png che rappresenta invece la nostra Texture.

ESERCIZIO 3

Stessa cosa dell’esercizio 2 ma utilizzando la sprite!

Alcuni punti:

- Come posso fare se non voglio che l’immagine risulti stretchata?
- Dalla classe Sprite ho la possibilità di cambiare pivot?
- Come posso fare se voglio che la sprite segua la posizione del cursore del mouse?
- Come posso ruotare la mia sprite?

AIV.FAST2D - WINDOW

Ecco alcune proprietà della Window:

- SetClearColor : setta il colore di default della finestra (quello che sta sotto tutto).

ViewPort: è dove la scheda video renderizza l'immagine (quindi non renderizza direttamente sul monitor). La ViewPort ha una posizione e una grandezza.
Di default la ViewPort ha la stessa grandezza della finestra e parte dal primo pixel in alto a sinistra.

- SetViewPort: con questo metodo è possibile cambiare la viewport della finestra!

ESERCIZIO 4

Partendo dalla base dell’esercizio 3, modificare il colore di “sfondo” della finestra, e la sua viewport.

ESERCIZIO 5

Attraverso l’utilizzo di una viewport, implementare uno “splitscreen”.