DEBUGGING E ALPHA BLENDING

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI

DEBUGGING

Cosa si intende per debugging?

Il debugging è una attività che consiste nell'individuazione e correzione (si spera) di uno o più errori (detti bug) nel software.

Il flusso solitamente dovrebbe essere questo:

- Il programmatore scrive il codice.
- Il programmatore valida il codice (con gli Unit Test che vedrete nei successivi anni).
- Il programmatore consegna il codice.
- Qualcuno testa il software e si accorge che ci sono molti errori.
- Il programmatore, dalla segnalazione degli errori del tester, deve capire dove sta il problema e risolverlo.

DEBUGGING - COME NON SI FA

Avete presente la funzione Console.WriteLine che ti permette di stampare a console qualcosa.

Ecco, la maggior parte dei neofiti lo utilizza come strumento di debug.

Non capisco se entro in quel punto oppure in quell'altro punto? Ma che problema c'è. Metto un bel Console.WriteLine con due output diversi e vedo a console cosa esce.

------LA MORTE DEL DEBUGGING------

IL DEBUGGER QUESTO SCONOSCIUTO

Praticamente ogni IDE esistente al mondo ormai implementa un debugger.

Cos'è?

Un fantastico amico che ti può aiutare a risolvere i bug della tua applicazione.

Cosa offre?

Offre diversi modi per vedere le operazioni eseguite dal codice DURANTE l'esecuzione.

DEBUGGER MAIN FEATURE

Ecco alcune feature:

- break point: ti permette di impostare dei punti in cui l'esecuzione si fermerà nel codice.
- timer su variabili per vedere quando cambiano valore
- esaminare percorso di esecuzione del codice

Attenzione che per poter usare il debuggare il software deve partire in modalità Debug.

DEBUGGER BREAK POINT

Per impostare un break point, basta cliccare sulla sinistra della riga in cui c’ è l’istruzione in cui si vuole fermare il flusso di esecuzione runtime del programma.

Una volta che il programma raggiunge quel punto (se durante l’esecuzione effettivamente raggiungerà mai quel punto) il programma si ferma e sarà possibile vedere il valore di tutte le variabili del nostro programma.

Da qui abbiamo diverse possibilità:

- Riprendere l’esecuzione in modo normale fino all’eventuale incontro di un nuovo break point.
- Procedere un'istruzione alla volta.
- Entrare dentro una funzione
- Uscire dalla funzione corrente
- Eseguire il codice fino alla riga in posizione del cursore
- Eseguire il codice fino a una riga selezionata

DEBUGGER ESERCIZIO

Scriviamo un programma potenzialmente buggato e analizziamolo con il debugger.

SPRITE e AIV.DRAW

Fino ad ora abbiamo visto come disegnare semplici forme andando a impostare un determinato colore di un determinato pixel.

Ovviamente questo metodo va bene se si vuole creare un gioco con forme molto semplici e colori “flat”.

Ma se io voglio poter renderizzare un personaggio che è stato disegnato precedentemente da un artista?

In gergo questo asset viene chiamato Sprite.

SPRITE e AIV.DRAW

Aiv.Draw espone una classe chiamata Sprite che ti permette di, passandogli il percorso di un'immagine, di estrarne la bitmap.

ATTENZIONE:
Fino ad ora abbiamo ragionato con colori con 3 canali, RGB.

Se noi gli diamo in pasto un .png, questo assets è un'immagine che ragiona con 4 canali: RGBA.

A, chiamato canale alpha, indica la "trasparenza" di quello specifico pixel.

Ma cosa vuol dire avere un pixel parzialmente trasparente?

ALPHA BLENDING

Significa che quel pixel ha un colore, con una certa trasparenza, e che quindi quando andrà renderizzato nella finestra, il colore del pixel della finestra sarà per una parte quello della sprite e per una parte il colore che già esisteva in quel pixel della finestra.

Questo procedimento si chiama alpha blending.

Algoritmicamente come si può ottenere questo risultato?

Partendo da questi parametri che possiamo ottenere dalla classe Sprite:

- height
- width
- bitmap

Andiamo a costruire la funzione DrawSprite nella classe GfxTools

IMPORTARE UN ASSET IN UN PROGETTO

Per creare un nuovo oggetto di tipo sprite, all'interno di Aiv.Draw, abbiamo bisogno di passargli una stringa che rappresenta il percorso del file dell'immagine da cui deve prendere tutte le informazioni (width,height e bitmap).

Una volta importato un asset nel progetto, come ci assicuriamo che, una volta buildato il programma, l'assets si troverà all'interno della nostra build?

DIFFERENZA TRA PATH ASSOLUTI E RELATIVI

Path assoluto: è il percorso completo nel file system di un determinato file. Generalmente non si usa perché se io ho il gioco installato in una certa directory e utilizzo il path assoluto di un certo asset, a quel punto se invece sposto la cartella in un altro punto del file system a quel punto l'asset non si troverà più in quel punto.

Dovrei fare in modo di poter dire, hey il mio file, a partire dalla posizione del mio progetto (dell'eseguibile) si trova in questa directory.

Path relativo: è il path del file system a partire dal path dell'eseguibile. Così se dico che un file si trova in Assets/alienBullet_0.png vuol dire che il percorso assoluto è quello dell'eseguibile + path relativo.

CLASSE SPRITE OBJ

Per comodità creiamo una classe che si chiama SpriteObj dove andiamo a memorizzare tutte le informazioni essenziali (compresa la reference alla Spite di Aiv.Draw) che ci possono permettere di renderizzarla.

CLASSE ALIEN BULLET

Ora andiamo a creare la classe alien bullet. Notiamo che abbiamo due immagini per i bullet. Questo vuol dire che possiamo creare una animazione dei bullet. Ma per ora usiamo solo l'alienBullet_0 e facciamo in modo di riuscire a far sparare i nemici.

Anche per questi bullet utilizzeremo la tecnica dell'ObjPooling.

DOMANDA: dove è meglio creare la nostra pool di bullet per gli alieni?

CREIAMO LA POOL IN ENEMY MGR

Per creare la pool dobbiamo:

- Creare un campo che rappresenta la pool di alieni.
- Durante la Init della classe dobbiamo creare tutti bullet
- Durante la Update dobbiamo fare l'update anche dei bullet
- Durante la Draw dobbiamo fare la Draw dei bullet
- Creare un metodo per ritornare un bullet dalla pool.

FACCIAMO SPARARE I NEMICI

Solo il primo alieno per colonna può sparare.
Andiamo nella classe Alien e prepariamo tutto per farli sparare :

- Creare due campi: CanShoot e nextShoot per contare quanto manca al prossimo colpo e nel costruttore settiamo nextShoot con un valore randomico.
- Nell'enemy mgr andiamo a settare CanShoot a un solo alieno per riga durante la Init.
- Nell'enemy mgr andiamo a creare un metodo Shoot per sparare.
- Quando un nemico viene distrutto viene passato il CanShoot al nemico prima sulla stessa colonna (se esiste).
- In Alien quando nextShoot raggiunge zero andiamo a sparare e impostiamo un nuovo valore randomico a nextShoot.