## Open Source 3D Engine

## OpenGL Rendering System

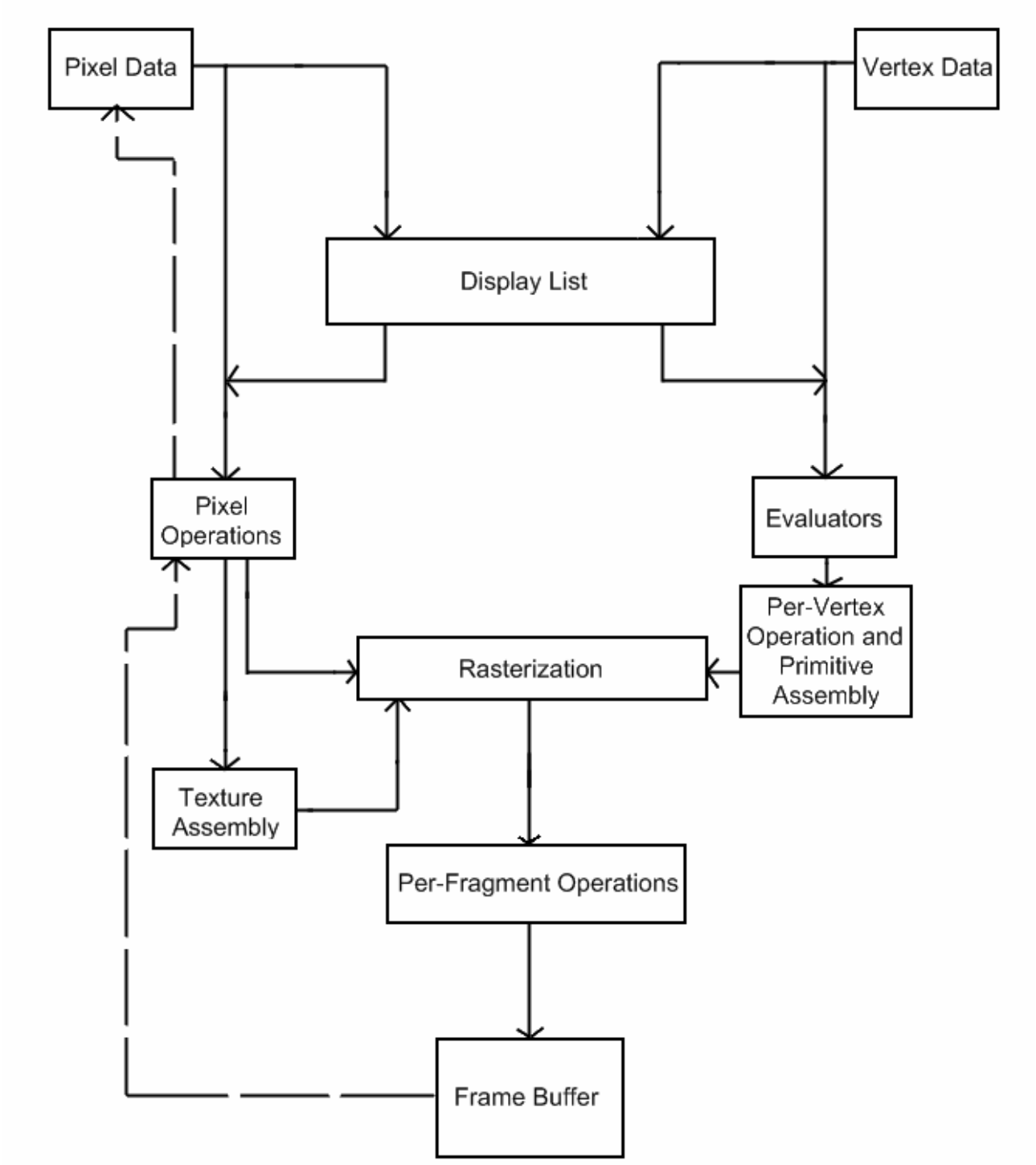
## Il Framework

I moderni mezzi di programmazione, consentono a noi sviluppatori di utilizzare librerie avanzate e testate che si prestano eccellentemente allo svolgimento delle funzioni che un software 3D deve avere.
La cosa che più mi ha spinto verso OpenGL è la sua portabilità su altri sistemi.
Creare un buon software, a mio avviso, non vuole dire solamente creare un programma avanzato, potente, preciso, ma anche un programma in grado di funzionare su tutte le piattaforme.
La attuale posizione nel mercato dei sistemi Linux, l'interesse della comunità informatica verso soluzioni alternative, sono tutte caratteristiche che vanno assolutamente tenute in considerazione nella progettazione di un software.
Ciò non toglie che qui, si fanno delle scelte non in base alle propie preferenze personali, ma in base al bacino d'utenza che un software in potenza potrebbe avere.
Benvenga quindi Windows, Linux e OpenGL che funge da collante frà i vari sistemi.

## GLUT

Grazie alla API GLUT, OpenGL Utility Toolkit, è possibile avvalersi di interessantissime e utilissime funzioni che, se da un lato sembrano "facilitare" lo sviluppo del framework, dall'altro offrono funzionalità che consentono di inizializzare con pochissimo codice l' API OpenGL.
In termini di praticità, questa è sicuramente è un'agevolazione non indifferente è non trascurabile.

## OpenGL Rendering Pipeline



Questa rappresentazione della pipelina OpenGL, riassume il funzionamento dell'API nella fase di elaborazione dei dati in input forniti dal programma.
I tipi di dati processati dalla pipeline, sono come si nota, Coordinate Spaziali(punti,vertici) e Pixels Data(Pixels, Immagini, Bitmaps).
Il processo di questo tipo di dati, avviene per vie differenti, ma il risultato finale è un insieme composto di dati che viene scritto nel Framebuffer, copiato nella memoria video ed inviato allo schermo. I vari stadi della pipelina compiono delle operazioni sui dati per definire l'assemblaggio finale dell'immagine.

Come si nota, all'inizio, il passaggio dei dati(PXd,VXd) non è vincolato al passaggio per le display list, ma queste comunque sono estremamente utili e costituiscono un modo per salvare i dati di un determinato oggetto e saltare alcuni passaggi della pipelina.

**Coordinate Spaziali(punti, vertici)**

I VXd, passano per gli evaluators, che compiono attraverso dei control points, ed a seguito di alcune operazioni di derivazione su questi, il lavoro di determinazione dei vertici delle curve Bezier o Nurbs.

Nel successivo stadio, Per-Vertex Operations, le coordinate di tutti i vertici vengono elaborate attraverso delle matrici di trasformazione che ne determinano effetti Traslazione, Rotazione, Ridimensionamento e vengono determinate le posizioni delle textures e gli effetti dell'illuminazione per-vertex.

Quindi, nella fase Primitive Assembly, avviene il Clipping dei Poligoni nascosti, i calcoli sulla profondità e le operazioni relative alla proiezione in prospettiva.

E importante notare che OpenGL puo' anche non lavorare con la proiezione in prospettiva, se viene inizializzata in maniera ortogonale.

**Pixels data**

Tutti i dati relativi alle immagini in input dal programma subiscono l'operazione di un-packing,ed in seguito dopo successive trasformazioni(rotazione-ridimensionameto) vengono passati o direttamente alla fase di Rasterization o alla Texture Assembly(responsabile dei calcoli sul texture mapping).

**VXd e PXd**

Nella fase Rasterization, i dati dei vertici e delle immagini si combinano e vengono trasformati in fragments.Ogni fragment rappresenta un pixel del Framebuffer, i poligoni vengono cosi' riempiti, viene applicato l'antialiasing, viene dato ad ogni fragment un colore ed una profondità.

Finite queste operazioni, vengono copiati sui fragments i pixels delle textures, viene effettuata la eventuale sovrascrittura o modifica dei colori dei poligoni, vengono effettuati test in base ai parametri sulla trasparenza ed in base ai valori contenuti nello stencil buffer o nel depth buffer.

A questo punto, avendo un Framebuffer pieno di tutti quei dati che servono alla visualizzazione finale, questo viene copiato nella memoria video e mostrato sullo schermo.

**Il motore 3d**

La creazione di un motore 3D, avendo bene a mente la OpenGL pipeline, è un'insieme di strutture dati e funzioni che, correlazionate con OpenGL e correlazionate trà loro, vengono a formare una complessa struttura di dati di input per la pipelina.

Caratteristiche tecniche attuali del progetto

Attualmente, il progetto è nelle sue fasi primordiali ed è scritto in liguaggio C. L'idea pero' di una conversione in C++ è sicuramente interessante e credo anche indispensabile sotto certi punti di vista.

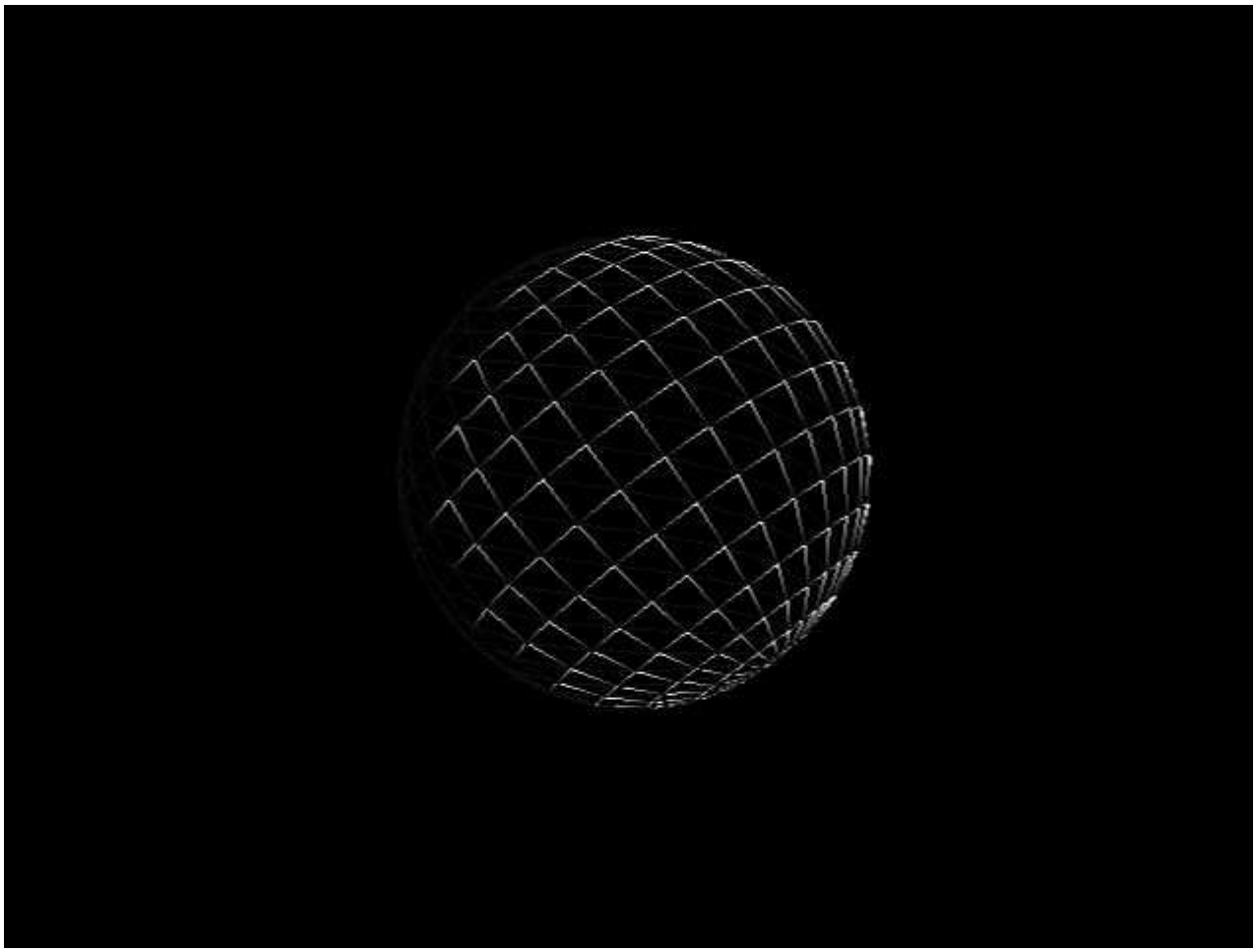
La struttura attuale è la seguente:

- ObjectModel responsabile delle definizioni delle strutture principali del motore come Vertex, Poligono, TextureCoords, Normals.
- Textures responsabile della funzione di caricamento delle textures
- Vectors responsabile delle funzioni di creazione vettore planare, determinazione della magnitude, normalizzazione della magnitude del vettore, prodotto scalare, prodotto vettore
- 3DSLoader responsabile della funzione di caricamento dei files di 3DStudio Max per l'importazione dei modelli 3D
- OGLManager responsabile di tutte le inizializzazioni della API OpenGL

Attualmente, le caratteristiche del programma sono le seguenti:

- Calcolo dell' illuminazione normale
- Texture Mapping
- Loader 3DSMax

La seguente preview mostra il sistema di illuminazione in azione



**Lo sviluppo del progetto – Il Team**

Per lo sviluppo concreto di questo progetto, sono necessarie diverse componenti, e soprattutto molto impegno, dato che non è un lavoro facile ne tantomeno breve.

La struttura del team che dovrebbe lavorare a questo software, deve comprendere diverse figure che essenzialmente sono:

- Sviluppatore Engine Editor
- Sviluppatore IA
- Sviluppatore Assembly
- Sviluppatore PhisicCore
- Analista-Debugger Multipiattaforma
- Modellatore 3D Studio Max
- Texture Artist

Sarebbe ben accetta inoltre una persona che abbia conoscenze di OpenGL ed in generale di grafica 3D che possa lavorare con me nello sviluppo del 3DCore, possibilmente di Roma.

Chiunque voglia partecipare, riceverà poi da me tutte le informazioni approfondite sul programma e tutti i chiarimenti del caso.

Mi auguro si possa concretizzare questa bellissima iniziativa, un ringraziamento a Melix per avere messo a disposizione questa forma di collaborazione ed un saluto a tutti quanti.

Andrea