HPMS Engine

# Concept

Partendo dal prototipo sviluppato tramite LWJGL (JRD3) verrà sviluppato un motore analogo più leggero e ottimizzato in codice nativo (C++)

# Filesystem

Per avere compressione, trasparenza e velocità di caricamento delle risorse, verranno utilizzati dei formati binari e un filesystem virtuale.

Utilizzando la libreria *PhysFS*, tutte le risorse presenti in un archivio (formato *7z*, in questo caso rinominato *hpak*) vengono caricate in fase di startup e montate su una directory virtuale strutturata nel modo seguente:

data  
|  
+--- Data.cfg  
|  
+--- scripts  
|  
+--- rooms  
|  
+--- shaders  
|  
+--- models  
|  
+--- textures  
|  
+--- screens  
|  
+--- sounds

Gli scripts, shaders, immagini e audio rimarranno in formato standard, mentre i modelli 3D e i descrittori delle ambientazioni verranno compressi in un formato binario.

# Engine

Il motore grafico sarà una versione ottimizzata e leggera di JRD3 (prototipo del medesimo in LWJGL3).

# Runtime

Il gameplay verrà realizzato interamente in LUA, per permettere di modificare e creare nuovi scenari di gioco senza modificare l’eseguibile.

Ogni scenario ha uno script associato, definito dalla seguente struttura:

*scene* = {  
 name = "Scene.01",  
 version = "1.0.0",  
 on\_init = **function**()  
 -- Eseguita in fase di inizializzazione.  
 **end**,  
 on\_input = **function**(key)  
 -- Eseguita in fase di inizializzazione.  
 **end**,  
 on\_loop = **function**()  
 -- Eseguita per ogni frame.  
 **end**,  
 on\_close = **function**()  
 -- Eseguita alla fine della scena.  
 **end**}

# Editor

L’editor verrà sviluppato in Java + JavaFX per velocizzare i tempi di realizzazione della GUI. L’applicativo permetterà di:

* Caricare la walkmap e le camere da un file esportato da blender;
* Caricare le immagini degli sfondi e le maschere di profondità;
* Poter visualizzare un’anteprima delle varie inquadrature con il proprio sfondo;
* Poter aggiuingere gli oggetti 3D della scena;
* Poter aggiungere comportamenti ai vari oggetti o alla scena stessa, tramite script LUA, usando un sistema di condizione/azione;
* Poter importare i vari modelli 3D e visualizzarne l’anteprima;
* Poter esportare dal progetto un HPAK contenente:
  + Un XML/JSON/BIN che descrive la scena (incluse camere, settori, scripts ecc…);
  + Le immagini ed eventuale audio;
  + I modelli 3D (che verranno prima convertiti nel formato binario usando una routine esterne scritta in C++, che importerà i modelli nelle strutture usate dal runtime e le serializzerà in relativi file binari).
* Poter salvare tutto il lavoro in formato *.hproj*