Documentación del motor:
Práctica final

Alberto Estirado López
Motores gráficos y plugins
Diseño y desarrollo de videojuegos 4.3

Introducción al motor:

El motor consta de un kernel y un scene manager. Es necesario inicializar el kernel y añadir las diferentes tasks. Las tareas se ejecutan en base a la current scene que tenga el scene manager. Las escenas se cargan en base a un xml. Tienen una lista de entidades y diferentes sistemas. Los sistemas se encargan de la utilidad de los componentes que son contenidos por las entidades.

En caso de querer añadir diferentes comportamientos, tenemos un controller system el cual se encarga de llamar a su update. Con esto podemos añadir los comportamientos que deseamos. Esto lo uso en la demo para hacer que el jugador se pueda mover con los inputs del teclado. Usamos el mismo sistema para las colisiones.

Carga de escenas con xml:

En el documento lo primero que se debe indicar es la scene, por lo que creamos un componente scene y podemos añadir un id, el cual funcionará como nombre de las scena. Cada escena está compuesta de un conjunto de entidades, es lo que definimos a continuación. Dentro del componente entities vamos definiendo entidades con cada uno de sus componentes.

Cada componente tiene un id, este lo usamos en la función load_scene() para crear los componentes adecuados. Todas las entidades deberían de tener un transform, es aquí cuando puedes definir un parent.Cada componente requiere cierta información para ser creado en la escena. Esta información puede dividirse en más componentes como puede ser en el caso del transform o un value como puede ser en el caso de mesh renderer.

Ejemplo:

```
<scene id="sample-scene">
        <entities>
      <entity id="player">
        <component id="transform">
          <position>
            <x>0</x><y>0</y><z>0</z>
          </position>
          <rotation>
            <x>0</x><y>0</y><z>0</z>
          </rotation>
          <scale>
            <x>0.8</x><y>0.8</y><z>0.8</z>
          </scale>
        </component>
        <component id="mesh_component">assets/models/sphere.obj</component>
        <component id="box_collider_component">
          <scale_x>1</scale_x>
          <scale_z>1</scale_z>
          <type>dynamic</type>
        </component>
      </entity>
        </entities>
</scene>
```