Proyecto

X-Kating

Grupo

WASTED HORCHATA

“Documento de definición del formato de niveles”

Hito: 2
Fecha entrega: 26-01-2018
Versión: 2

Componentes:

- Luis González Aracil
- Laura Hernández Rielo
- Adrián Francés Lillo
- Pablo López Iborra
- Alexei Jilinskiy

1. Contexto de juego

Nuestro proyecto implementa una serie de niveles donde transcurrirá la totalidad de la acción. En estos niveles se deberán cargar múltiples objetos con variables de posición, tamaño, modelo, texturas, etc…

Implementamos para la generación de objetos una arquitectura basada en componentes. Esta arquitectura nos permite una mayor flexibilidad en el diseño, definiendo el comportamiento de un objeto en base a sus componentes.

2. Formato a utilizar

El formato elegido para la carga de niveles ha sido XML (eXtensible Markup Language). Es un meta-lenguaje que permite almacenar datos de manera legible, de carga rápida que nos permite definir una estructura jerárquica de objetos, a los cuales se les podrán asignar diferentes componentes.

Esta jerarquización permite componer objetos con otros, y generar estructuras complejas de objetos. Esta característica en especial nos permite diseñar esqueletos de animación acoplables y con múltiples huesos.

3. XML

La estructura básica de un nivel en XML consistirá en las variables específicas del nivel, tales como su nombre y sus posiciones iniciales, y a continuación la lista completa de todos los nodos de mapa.

Los nodos de mapa vendrán definidos por su nombre, el nombre de su mapa previo y su tipo.

Cada nodo de mapa contendrá la información específica de una zona concreta del mapa. A tener en cuenta, aquí se contendrá la lista de las cajas de colisión de esa zona del terreno. Estos nodos de mapa estarán definidos como componentes del nivel.

Finalmente, aparecerán añadidos todos los objetos presentes en la pista, tales como las cajas de los objetos o las rampas de aceleración, con la información específica de cada uno.

Los objetos serán identificados con la etiqueta objeto, y contendrán dos variables, “id” y transformación inicial.

Un objeto será contenedor de múltiples componentes. Cada componente que contenga un objeto deberá ir definido por la etiqueta “component”. Cada componente contendrá de forma obligatoria la etiqueta “name”, conteniendo el nombre o tipo de componente. Dependiendo del tipo de componente, éste contendrá diferentes variables que lo definan.

4. Estructuras predefinidas

1. **Nivel:** Se le añadirá una id propia a cada nivel, además contiene un atributo “name” donde se le pasará un nombre único para cada nivel. Dentro de éste, estarán contenidos todos los elementos del nivel como el mapa, posiciones iniciales..

```
<level id=”0000” name=”NeoCityRampage”>
</level>
```

2. **Objeto:** Se le pasará una id propia al objeto, además de su posición. Internamente contendrá las componentes necesarias para el propio objeto

```
<object id=”1” pos=”0,0,0” rot=”0,0,0” sca=”0,0,0”>
</object>
```

3. **Componente:** Contendrá un atributo “name”, donde se pondrá el nombre del componente a crear. Según el tipo de componente, además se le añadirán sus variables propias necesarias para la inicialización de dicha componente.

```
<component name=”camera” targetId=”1” />
```

4. **Posición inicial de cada jugador:** Se creará una lista de posiciones iniciales válidas para el mapa. Posteriormente se añadirán posiciones, se les pasará por parámetro su posición inicial.

```
<position>
    <place position=”1” pos=”0,0,0” />
    <place position=”2” pos=”20,20,0” />
</position>
```

5. **Caja de colisión:** Las colisiones con el terreno vendrán definidas por trapezoides. Se le pasaría como parámetro los cuatro puntos de los vértices, de manera que se podrán crear los planos deseados.

```
<bbox p1=”0,0,0” p2=”0,0,0” p3=”0,0,0” p4=”0,0,0” friction=”1” />
```

5. Blender y Wavefront

El formato XML anteriormente descrito será formado a partir de la lectura de un archivo Wavefront (.obj) generado preferentemente con Blender, aunque en principio debería ser posible utilizar cualquier programa de modelado 3D siempre y cuando pueda exportar a formato Wavefront. Dicho fichero es posteriormente interpretado por un programa realizado en Python, de modo que se traduce a un XML que será leído por el juego.

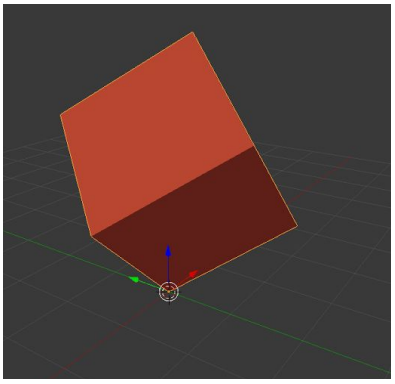
Para ejecutar el parser se debe hacer lo siguiente:
Desde la carpeta Game, ejecutar en terminal

python ../Parser/pythonParser.py ruta-al-obj-a-leer

El resultado de la lectura de fichero será colocado en Game/media/xml con el nombre de 'circuit.xml'.

Para crear distintos elementos en Blender y que sean interpretados correctamente se debe seguir una serie de pautas:

IMPORTANTE: Para marcar la posición de los objetos, luces, etc se utilizará un cubo. Se les podrá dar colores libremente para diferenciar unos tipos de otros, pero, es muy importante que se utilice el cubo que se da en la plantilla por defecto de Blender para marcar la posición, ya que si no se corre el riesgo de que el parser interprete mal la posición. La posición del objeto viene denotada por el vértice inferior del cubo.



IMPORTANTE: A excepción del terreno lógico, cualquier otro objeto podrá ser escalado y rotado en escena (esto permite una mejor reutilización de los meshes). Para escalar o rotar objetos se añadirá al final del nombre de cada objeto una secuencia del tipo **_RotX:RotY:RotZ_ScaX:ScaY:ScaZ**. Si no se incluyen se asumirá un valor 0. Se puede incluir solo la rotación, pero la escala no se puede incluir sin añadir la rotación previamente. La rotación y la escala se indican de este modo por **el formato Wavefront no guarda la rotación ni escala de un objeto**, solo la posición absoluta de sus vértices.

-Terreno: El nombre del objeto debe empezar por **'T_'.** Debe ser un solo objeto, formado por trapezoides. Es muy importante para el correcto funcionamiento del terreno que los cuatro puntos que forman los trapezoides sean coplanares. Cabe destacar que se crea un gameObject por cada terreno existente.

El espacio de id reservado para trozos de terreno va de **0 a 9999**.

-Waypoints: El nombre del objeto debe ser **W_Nivel_Radio_Rotación_Escala.** Para los waypoints se recomienda preferentemente utilizar cubos de color **negro**.

El espacio de id reservado para waypoints va de **10000 a 13999**.

-Luces: El nombre del objeto debe ser **L_ID_TipoDeLuz_Radio_Rotación_Escala.** El tipo de luz por el momento podrá ser "P" (luz puntual) o "D" (luz direccional). Por ejemplo "L_0_P_60.5" crea una luz en la posición de nuestro cubo, creando una luz puntual con un radio de 60.5, con rotación y escala 0 por defecto. Las Ids pueden ir de 0 a 1999. **Importante:** la ID final del objeto puede ser distinta de la asignada en Blender, ya que se hace de forma automática; sólo es necesaria porque Blender no permite nombres de objetos repetidos. Para las luces se recomienda preferentemente utilizar cubos de color **verde**.

El espacio de id reservado para luces va de **14000 a 15999**.

-Cajas de objetos: El nombre del objeto debe ser **I_ID_Rotación_Escala.** La ID, como para las luces, solo sirve para Blender, pero no tiene efecto en el juego. Para los waypoints se recomienda preferentemente utilizar cubos de color **amarillo**.

El espacio de id reservado para cajas va de **16000 a 17999**.

-Rampas: El nombre del objeto debe ser **R_ID_Velocidad_TiempoConstante_TiempoCaída_Rotación_Escala.** La ID, como para las luces, solo sirve para Blender, pero no tiene efecto en el juego. Para las rampas se recomienda preferentemente utilizar rectángulos de color **rosa**.

El espacio de id reservado para rampas va de **18000 a 19999**.

-Objetos modelados: El nombre del objeto debe ser **O_ID_NombreFichero_RadioColisión_Altura_Rotación_Escala.** La ID, como para las luces, solo sirve para Blender, pero no tiene efecto en el juego. Para el nombre del fichero, el mesh debe estar colocado dentro de una carpeta con su mismo nombre dentro de media/mesh/. Para el radio de colisión, si es 0 no se asignará componente de colisión. Para los modelados se recomienda preferentemente utilizar cubos de color **naranja**.

El espacio de id reservado para modelos va de **20000 a 23999**.

-Línea de salida: El nombre del objeto debe ser **S_ID_Rotación_Escala.** La ID, como para las luces, solo sirve para Blender, pero no tiene efecto en el juego. Para la linea de meta se recomienda preferentemente utilizar rectángulos de color **rojo**.

El espacio de id reservado para líneas de meta va de **24000 a 24999**.

-Nota: A partir de **25000** el espacio disponible restante está reservado para personajes. Hay más de 30000 espacios disponibles, por lo que futuros tipos de objeto se podrían colocar sin problemas.

6. Apéndice 1: Formato de ejemplo del nivel

Nivel

```
<!-- Initial variables →
<position>
        <place position=”1” pos=”0,0,0” />
        <place position=”2” pos=”20,20,0” />
        <place position=”3” pos=”40,0,40” />
</position>

<!-- Node map list →
<object id=”1000” pos=”0,0,0” rot=”0,0,0” sca=”0,0,0”>
        <component name=”terrain” left=”-1” right=”-1” top=”-1” bot=”-1”>
                <bbox p1=”0,0,0” p2=”0,0,0” p3=”0,0,0” p4=”0,0,0” friction=”0.2” />
        </component>
        <component name=”objectRender” mesh=”assets/box.obj />
</object>
<object id=”1001” pos=”0,0,0” rot=”0,0,0” sca=”0,0,0”>
        <component name=”terrain” left=”-1” right=”-1” top=”-1” bot=”1000”>
                <bbox p1=”0,0,0” p2=”0,0,0” p3=”0,0,0” p4=”0,0,0” friction=”0.1” />
        </component>
        <component name=”objectRender” mesh=”assets/box.obj />
</object>

<!-- Prize boxes list →
<object id=”2000” pos=”0,0,0” rot=”0,0,0” sca=”0,0,0”>
        <component name=”move” mass=”1” />
        <component name=”collision” kinetic=”false” type=”default”>
                <bbox p1=”0,0,0” p2=”0,0,0” p3=”0,0,0” p4=”0,0,0” friction=”1” />
        </component>
        <component name=”objectRender” mesh=”assets/box.obj />
</object>
<object id=”2001” pos=”0,0,0” rot=”0,0,0” sca=”0,0,0”>
        <component name=”move” mass=”1” />
        <component name=”collision” kinetic=”true” type=”itemBox” radius=”2” />
        <component name=”objectRender” mesh=”assets/box.obj />
</object>
```