BARX ENGINE (v 1.0)

Motores Gráficos y Plugins

26 / 01 / 2020

Jorge Bárcena Lumbreras

ÍNDICE

GUIA DE INSTALACIÓN 3

GENERACÓN DE UNA ESCENA A PARTIR DE UN ARCHIVO XML 6

COMPONENTES ESPECIALES 10

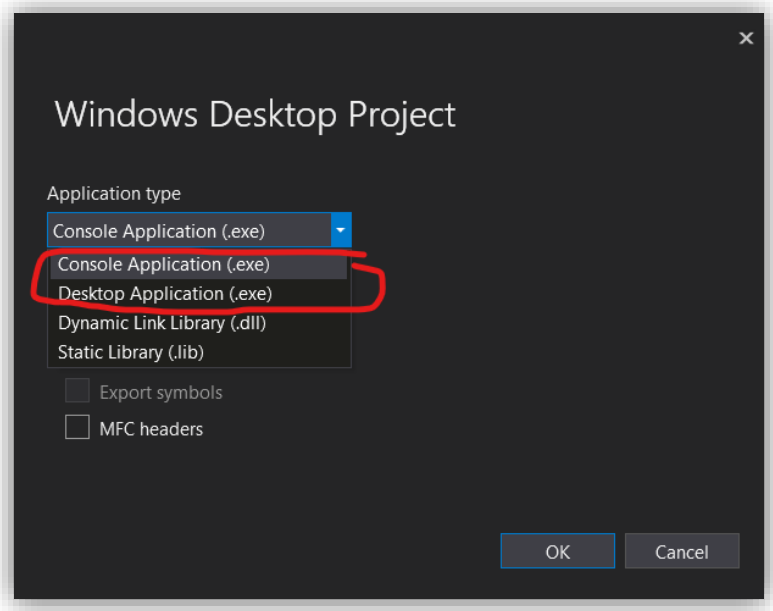
BControlComponent **10**

DEMO 11

GUIA DE INSTALACIÓN

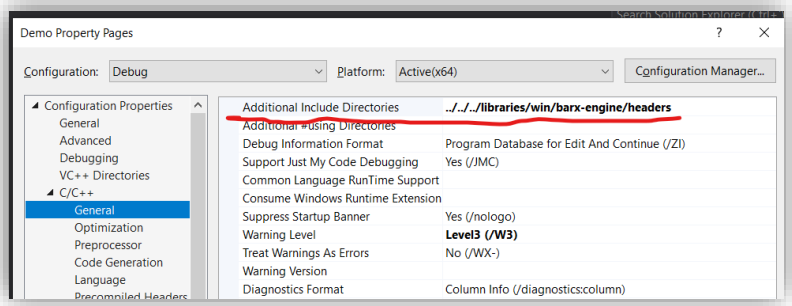
Para poder hacer uso del motor BarxEngine, se deberán incluir los siguientes archivos en la configuración del proyecto. Todos los archivos necesarios para el uso de la librería se encontrarán en la carpeta "*binaries*", incluida en el paquete descargado. Los pasos a seguir son los siguientes:

1. **Se creará un proyecto de aplicación en visual Studio. Elegiremos la configuración que mas se adapte a nuestro proyecto. Se recomienda crear un proyecto vacío.**



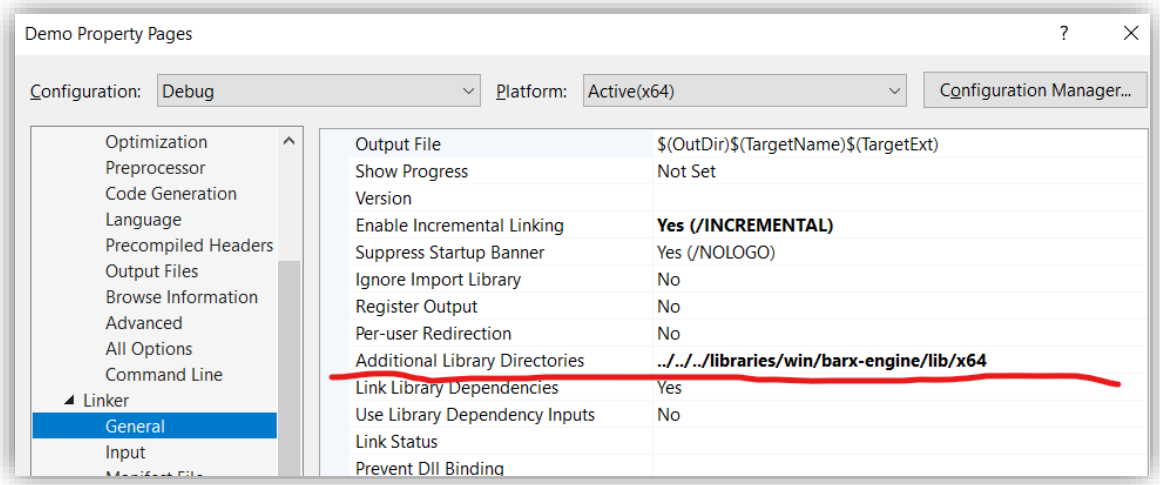
2. **Crearemos un archivo .cpp (Para que el proyecto sea configurado con las opciones de C++), e incluiremos los archivos ubicados en "*binaries/librerias estaticas/barx-engine/headers/*" en el apartado de *"Project properties -> C++ -> General -> Additional includes directories".***

3. **Ahora**

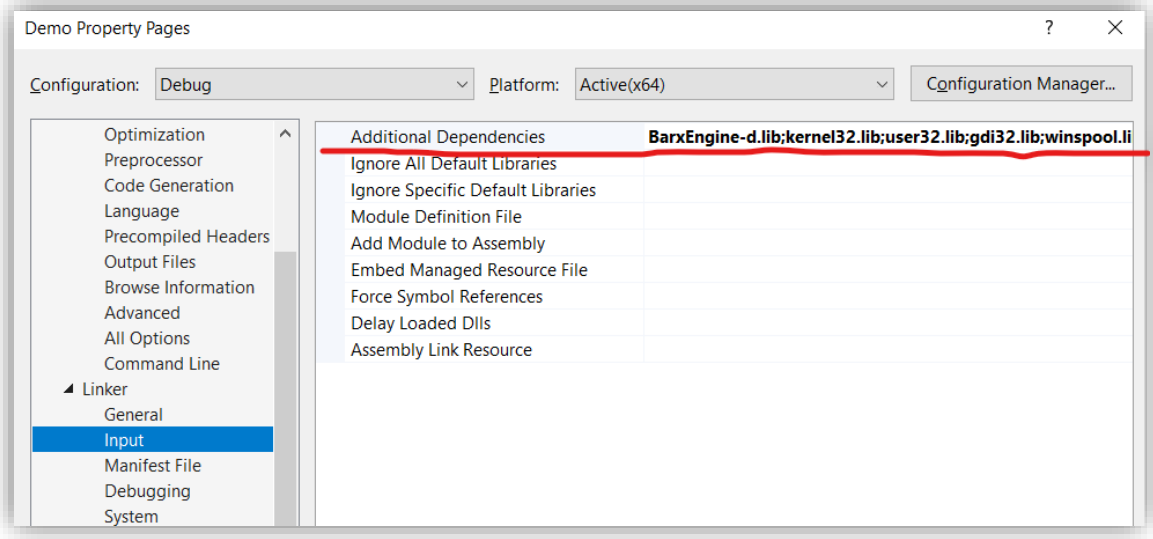


incluiremos la carpeta donde está contenida la librería estatica (.lib) del motor, ubicada en "*binaries/librerias estaticas/barx-engine/lib/x64/*"

en el apartado de *"Project properties -> Linker -> General -> Additional libraries directories"*.



4. **Por último, debemos incluir el archivo .lib, que se encuentra en "*binaries/librerias estaticas/barx-engine/lib/x64/*" en el apartado de "*Project properties -> Linker -> General -> Additional libraries directories*". En este paso veremos que hay dos archivos en la carpeta *"x64"*, cada uno de estos archivos esta precompilado en función de la configuración que funcionará. El archivo que tiene el sufijo "-d" está compilado en configuración de debug, y el archivo con el sufijo "-r" está compilado en configuración de release.**



5. **Para que el proyecto se pueda ejecutar, de deberán copiar los archivos contenidos en la carpeta de "*binaries/ librerias dinámicas*" deberán estar copiadas en el directorio donde se encuentre la solución de nuestro proyecto, ya que sin esos archivos, librerías de terceras utilizadas en el motor como SDL2 o SDL2Mixel, no pueden ser ejecutadas correctamente.**

GENERACÓN DE UNA ESCENA A PARTIR DE UN ARCHIVO XML

Para cargar una escena debemos crear un archivo .xml donde se crearán las entidades que aparecerán en la escena y los componentes que tendrán cada una. Una vez cargado este archivo podremos ejecutar las escena llamando al método run() del objeto de tipo escena. Hay unas consideraciones especiales que tenemos que tener en cuenta antes de crear un archivo de este tipo. Todas las entidades deberán de tener como primer componente un TransformComponent, que determinará en que posición de la escena se encuentra dicha entidad.

A continuación, se incluye un ejemplo de cómo se podría construir una escena:

```
<?xml version="1.0"?>
<scene WindowName="Demo - BarxEngine" windowWidth="800"
windowHeith="600" fullScreen="0">

  <entity id="Camera">
    <component type="BTransform_Component" >
      <position>0,1,5</position>
      <rotation>0, 0, 0</rotation>
      <scale>1,1,1</scale>
    </component>
    <component type="BCameraComponent">
      <fov>20</fov>
      <near>1</near>
      <far>50</far>
      <aspectRatio>1</aspectRatio>
    </component>
    <component type="BControlComponent"/>
  </entity>

  <entity id="Light">
    <component type="BTransform_Component" >
      <position>10,10,10</position>
      <rotation>0, 0, 0</rotation>
      <scale>1,1,1</scale>
    </component>
    <component type="BLightComponent"/>
  </entity>

  <entity id="Player">
    <component type="BTransform_Component" >
      <position>0, -2, -2</position>
      <rotation>0, 0, 0</rotation>
      <scale>0.05,0.05,0.05</scale>
    </component>
    <component type="BShereColliderComponent">
      <radius>0.1</radius>
    </component>
    <component type="BRenderObjectComponent" >
```

```
      <model>../../../binaries/assets/models/spider-
man.obj</model>
    </component>
    <component type="BCharacterController">
      <speed>5</speed>
      <Up>W</Up>
      <Down>S</Down>
      <Left>A</Left>
      <Right>D</Right>
    </component>

  </entity>

  <entity id="Floor">
    <component type="BTransform_Component" >
      <position>+2, -2, -1</position>
      <rotation>0, 0, 0</rotation>
      <scale>16,2,10</scale>
    </component>
    <component type="BRenderObjectComponent" >
      <model>../../../binaries/assets/models/plane.obj</model>
    </component>
  </entity>

  <entity id="Wall1">
    <component type="BTransform_Component" >
      <position>0, -2, -10</position>
      <rotation>0, 0, 0</rotation>
      <scale>5,1,1</scale>
    </component>
    <component type="BRenderObjectComponent" >
      <model>../../../binaries/assets/models/wall.obj</model>
    </component>
    <component type="BBoxColliderComponent" >
      <max>16,0,2  </max>
      <min>16,0,0</min>
    </component>
  </entity>

  <entity id="Wall2">
    <component type="BTransform_Component" >
      <position>-12.3, -2, 0</position>
      <rotation>0, 90, 0</rotation>
      <scale>8,1,1</scale>
    </component>
    <component type="BRenderObjectComponent" >
      <model>../../../binaries/assets/models/wall.obj</model>
    </component>
    <component type="BBoxColliderComponent" >
      <max>1,2,16</max>
      <min>0,0,16</min>
    </component>
  </entity>

  <entity id="Wall3">
    <component type="BTransform_Component" >
```

```
    <position>4, -2, 10</position>
    <rotation>0, 0, 0</rotation>
    <scale>5,1,1</scale>
  </component>
  <component type="BRenderObjectComponent" >
    <model>../../../binaries/assets/models/wall.obj</model>
  </component>
  <component type="BBoxColliderComponent" >
    <max>14,0,2  </max>
    <min>14,0,0</min>
  </component>
</entity>

<entity id="Wall4">
  <component type="BTransform_Component" >
    <position>+16.3, -2, 0</position>
    <rotation>0, 90, 0</rotation>
    <scale>8,1,1</scale>
  </component>
  <component type="BRenderObjectComponent" >
    <model>../../../binaries/assets/models/wall.obj</model>
  </component>
  <component type="BBoxColliderComponent" >
    <max>2,0,0</max>
    <min>-11,1,20</min>
  </component>
</entity>

<entity id="Enemy1">
  <component type="BTransform_Component" >
    <position>-11,-2,8</position>
    <rotation>0, 0, 0</rotation>
    <scale>0.01,0.01,0.01</scale>
  </component>
  <component type="BShereColliderComponent">
    <radius>0.1</radius>
  </component>
  <component type="BRenderObjectComponent" >
    <model>../../../binaries/assets/models/pig.obj</model>
  </component>
  <component type="BControlComponent"/>
</entity>

<entity id="Enemy2">
  <component type="BTransform_Component" >
    <position>11,-2,8</position>
    <rotation>0, 0, 0</rotation>
    <scale>0.01,0.01,0.01</scale>
  </component>
  <component type="BShereColliderComponent">
    <radius>0.1</radius>
  </component>
  <component type="BRenderObjectComponent" >
    <model>../../../binaries/assets/models/pig.obj</model>
  </component>
  <component type="BControlComponent"/>
```

```
  </entity>

  <entity id="Enemy3">
    <component type="BTransform_Component" >
      <position>-11, -2, -8</position>
      <rotation>0, 0, 0</rotation>
      <scale>0.01,0.01,0.01</scale>
    </component>
    <component type="BShereColliderComponent">
      <radius>0.1</radius>
    </component>
    <component type="BRenderObjectComponent" >
      <model>../../../binaries/assets/models/pig.obj</model>
    </component>
    <component type="BControlComponent"/>
  </entity>

  <entity id="Enemy4">
    <component type="BTransform_Component" >
      <position>11, -2, -8</position>
      <rotation>0, 0, 0</rotation>
      <scale>0.01,0.01,0.01</scale>
    </component>
    <component type="BShereColliderComponent">
      <radius>0.1</radius>
    </component>
    <component type="BRenderObjectComponent" >
      <model>../../../binaries/assets/models/pig.obj</model>
    </component>
    <component type="BControlComponent"/>
  </entity>

</scene>
```

En este ejemplo podemos ver como cada componente tiene una serie de parámetros que se pueden configurar, es necesario que todos los parámetros estén especificados en el archivo XML, para la correcta ejecución de la escena.

COMPONENTES ESPECIALES

Hay una serie de componentes los cuales podremos utilizar desde el proyecto que creemos. A continuación, se detallan cuales son esos componentes y como podemos acceder a ellos y utilizarlos de manera correcta.

BControlComponent

Este componente permite ejecutar una función que hayamos creado previamente en nuestro proyecto, durante el update del juego. Para asignar la función que queremos que se ejecute en el update, lo debemos hacer de la siguiente forma:

```
scene->getEntity(Id)->getComponent<BControlComponent>()-setFunction(MyFunction);
```

En este caso la función "MyFunction" deberá tener dos parámetros:

- float time → Tiempo que ha pasado desde el anterior fotograma.
- shared_ptr<BEntity> Entity → Entidad a la cual está asignada esa función.

BColliderComponent

Este componente permite ejecutar una función que hayamos creado previamente en nuestro proyecto, cuando la entidad que la contiene colisiona con otra entidad. Para asignar la función que queremos que se ejecute, lo debemos hacer de la siguiente forma:

```
scene->getEntity(Id)->getComponent<BColliderComponent>()-setFunction(MyFunction);
```

En este caso la función "MyFunction" deberá tener dos parámetros:

- shared_ptr<BEntity> A → Entidad que contiene la función.
- shared_ptr<BEntity> B → Entidad con la que se ha generado la colision.

DEMO

En el descargable hay una demo, que consta de las siguientes características:

1. **Se crea una escena con un suelo y 4 paredes.**
2. **Un personaje controlable con el CharacterControllerComponent.**
3. **4 enemigos situados en las esquinas que con el componente BControlComponent, buscan al jugador y lo persiguen.**
4. **Cuando se genera una colisión entre dos entidades, suena un sonido de aviso.**
5. **La cámara sigue al jugador durante todo el tiempo.**