IDentity Engine

Serialization Format

By team

Looking For Group (LFG)
Filipe Antunes Da Silva, Lilian Gadeau and Gabriel Meloche

May 19, 2020

Scene

Every GameObject is first serialized with its name and its variables, then all of its Components are serialized individually with the information necessary to rebuild them as they were. Every Component is contained within brackets (“{}”).

To declare a new GameObject, the keyword GAMEOBJECT is recognized by the parser, and the parser looks for a “;” to know when a GameObject declaration ends.

If a GameObject contains a nested type, its values are directly written into further indented brackets.

If a variable is a mathematical type (vectors, matrices, etc.) that contains more than one primitive type, the primitive variables are written sequentially. For example, a zero-Vector3 would be written as:
VariableName 0 0 0.

Each GameObject is serialized as follows:

```
GAMEOBJECT
GameObjectName
m_isActive Value
class ComponentClass
{
        VariableName1 Value1
        VariableName2 Value2
        …

        MathematicalVariableName Value1 Value2 Value3…

        NestedVariableName
        {
                VariableName1 Value1
                VariableName2 Value2
                ...
        }

}
;
```

ResourceManager

Resources included in our first beta build include Models, Textures and Materials. Each resource is declared with its own text line, starting with its type keyword amd followed by its respective information needed to reload it during the engine's start.

Models need their relative path followed by their name. Same goes for textures. Materials need their name, the pixel shader and the vertex shader they will use, and finally their associated texture's name. For now, only our default shaders can be used.

Resource Manager serialization goes as follows:

```
MODEL ModelPath ModelName
TEXTURE TexturePath TextureName
MATERIAL MaterialName PSName VSName TextureName
```