Rendering Order Optimization Comparative

Index

Context

Hello! We are Lucho Suaya and Roger Leon, two students of Videogames' Design & Development career of Universitat Politècnica de Catalunya, and we are developing a 3D Game Engine for Engines subject, taught by Marc Garrigó, called Cronos Engine, and we are one of the teams in charge of building a Shaders Sub-System to use in Project III subject of the next semester.

In the development transcourse, building the rendering pipeline, we came across with a doubt: How do we render as fast as we can?
Having into account the importance of speed and performance in a context as a 3D Game Engine, the purpose of this document is to make a comparative between different ways to render in a 3D Game Engine to test them and see which one is faster. This will be expanded in the next sections.

Cronos Engine Rendering Pipeline

In order to render objects in Cronos Engine, we have built a Mesh System. Each rendered object has a mesh component storing a mesh - understood as a pile of vertices (we won't go deeply explaining engine's foundations such as how meshes and geometry work, we will assume you have some knowledge on it, anyway, is not important to fully know about it) - and each mesh will call a function in render that stores in a list the game objects that must be rendered. Then, in the Renderer Post-Update, this list is accessed to draw each mesh.
However, in this process, intervenes another system, since we need and we want to draw things with colors and textures, we have also made a material system, so when we want to draw in the Post-Update of the Renderer, we pick the material that the game object has (if there is none, we use a default one) and we bind it before binding the Vertex Array Object (VAO) to draw and finally unbind both elements.
In addition, since we are using a Shader System, this means that we need to previously bind it and also send some data to it so we can render properly.

So, to conclude, as told in the Context section, when we were building this Renderer System, we start discussing which was the best way to do all this stuff as fast as possible and with the methods and tools we knew, since the material needs to share considerable information with the shader (textures, colors,...), and the shader needs other information a part of the one provided by materials (camera position, for instance), we thought in four different ways to do it, and our objective with this, is to measure them and use the better one (which in our case will be the fastest).

Shader Data

Before jumping head first to explain how everything will go, we need to know which information a shader needs to render. For our purposes, assume that only a default shader is being used for all rendered objects, to avoid external interference that may complicate more than contribute. This default shader needs the next information (passed as "uniforms" variables; their GLSL data type is specified in parenthesis):

- Camera Matrices (view and projection - mat4) - Used to render in a defined position. They are multiplicated, so they can be passed one by one with two uniforms and multiplicate them in the shader (which is our case) or with the multiplication already done in CPU and with one uniform.
- Camera Position (vec3) - Mainly to calculate lighting. As the name indicates, depends on the camera.
- Model Matrix (mat4) - Object's transformation matrix. A bit the same than the previous ones, the difference is that the other ones depend on the rendering camera, and this one on the object, so it has to be passed per object drawn.
- Material Ambient Color (vec4 or vec3) - Passed for each material to calculate final fragment color.

- Diffuse and Specular Textures (sampler2D) - Passed for each material to calculate final fragment color.
- Material Shininess (float) - The same than the previous two.

With all this data we have enough. There can be more data, of course, in our case we have other data, mainly to do lighting calculations or Z-Buffer rendering, but they are not important for the purpose of this document, so we will leave them apart. Moreover, they are passed with each light (except Z-Buffer Rendering, which depends on Renderer), they would add undesired complication to all this since they must be passed always from renderer, so we will do as if they don't exist, since they will be and behave the same for all cases we want to measure. You might think the same for the camera, and you would be right, but the camera is necessary to render (Z-Buffer and lighting aren't), therefore we want to use them here to show realistically how they affect depending on the binding/data-sending order.

So, as far as we concern, we see three types of data (and this is not a surprise, it was a bit explained in the previous section), the data depending on each material (like textures, color, shininess), the data depending on each object (model matrix) and the external data concerning the camera. From here, we see four possible lines of action, again, explained in the next section.

Rendering Cases

As previously told, we differentiate four ways to order all the data explained, and we will explain the theoretical approach here. With no more delay, let's explain the cases.

Case 1 - Brute Force

This case will be the most basic one. All the objects to render in the list will be traversed and each one will bind the referenced material it has, and, at each material bind, all the data will be passed to the shader. With this, the pipeline would be:

1. Bind Shader
2. Run through the objects to render list
3. For each object to render, get its referenced material and bind it
    a. At each material bind, pass all the needed uniforms/data (camera, object and material data) to the shader and bind the needed textures
    b. Bind VAO and draw
    c. Unbind object's material (which will unbind the textures) & VAO
4. Once finished the loop, Unbind Shader

As told, this is the simpler one and is expected to be the slower, too.

Case 2 - Divide et Impera

This will be more or less the same, but we will divide the data passed to the shader, so we will send the general data (the camera one) in the Renderer, only once, before going through the rendering object's loop, instead of passing it at each object's material bind. The material and object's data will be passed in the material bind (remember that each object binds its referenced material, so, with each object, a material is bound/unbound to draw). The pipeline would be:

1. Bind Shader
2. Pass general data (camera one)
3. For each object to render, get its referenced material and bind it
    a. At each material bind, pass the remaining needed uniforms/data (object and material data) to the shader and bind the needed textures
    b. Bind VAO and draw
    c. Unbind object's material (which will unbind the textures) & VAO
4. Once finished the loop, Unbind Shader

This seems to be a very logical one. Expects to be better than the first case, and it seems to be the one that is faster to implement in function of its performance (we will get the results later and we'll see it!).

Case 3 - Divide et Impera with an Ordered Vector by Materials

For this case we will need an algorithm or a function that orders the objects to render in function of the material they have, so, if it's expected to be the faster case, it might not be since we can lose performance with that algorithm.
So the idea is that, if two or more objects share the same material, they have to be aside in the objects to render vector.Then, first bind the material and then run through all objects with that material to draw them without binding/unbinding the material at each iteration. This way, we can possibly win some performance, but we have to test if this whole performance is not lost in the ordering step.
So, the pipeline would remain as (we will need here the help of a new bool called *changeMaterial*):

1. Order the objects to render list, put the objects with the same material together
2. Bind Shader
3. Pass general data (camera one)
4. Set *changeMaterial* to false
5. For each object to render
    a. If the material ID is not the same than the next one, set *changeMaterial* to true
    b. Bind VAO, pass object data and draw
    c. Unbind VAO
    d. If *changeBool* is true, set it to false. Unbind object's material (which will unbind the bound textures) and bind the material of the next object, pass the

remaining needed uniforms/data (material data) to the shader and bind the needed textures

6. Once finished the loop, Unbind Shader and the last bound material.

This is expected to improve the performance, but it has to be tested to prove it.

Tests & Measurements

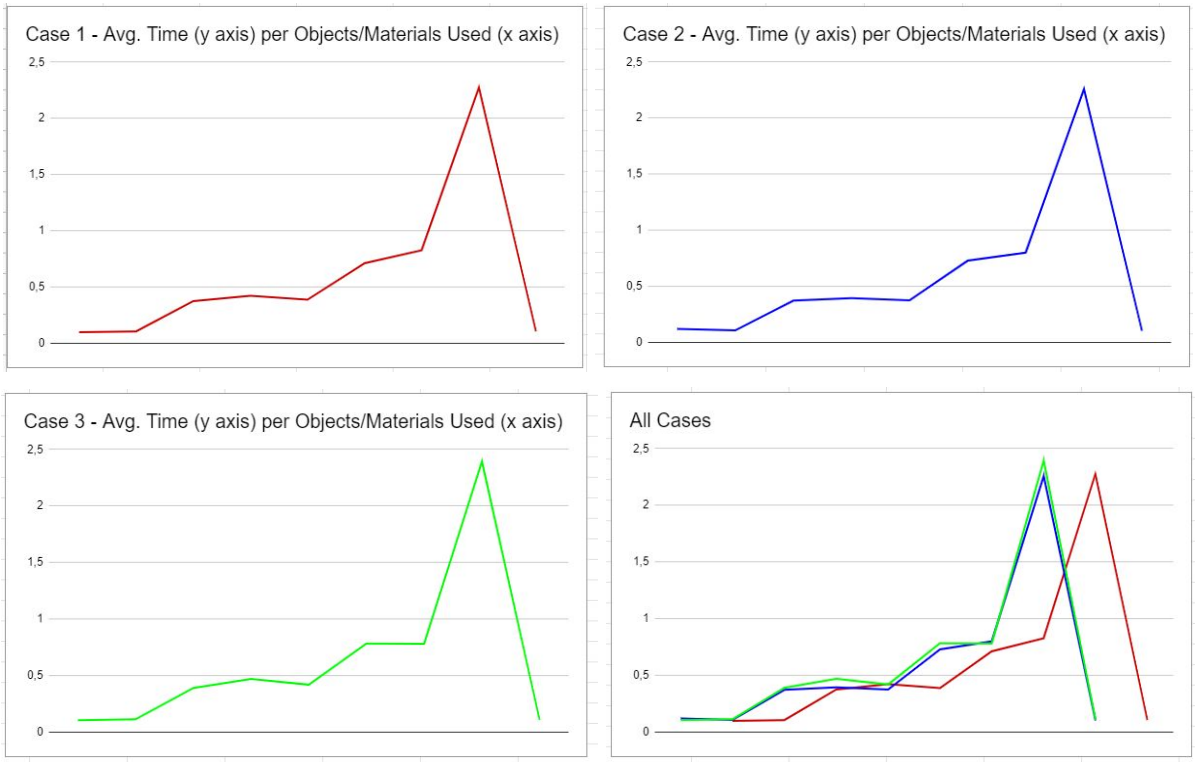
To properly perform the tests and measurements, we will bring each case under different situations:

- Using 1 Game Object (a sphere primitive) with the default material
- Using 1 Game Object with a different material than the default one
- Using 5 Game Objects (primitives) with the default material
- Using 5 Game Objects, with different materials each
- Using 5 Game Objects with the same material, different than the default one
- Using 10 Game Objects (primitives) with the default material
- Using 10 Game Objects with the same material, different than the default one
- Using a Street Model already built, having each object different materials but some of them the same than others.
- A high poly model already built, with different objects and materials.

They will all be in the same position and context (no lights in scene, no cameras rather than the engine's one…) and, for each case, we will use the same materials for the game objects used, to lower the error margin in calculations. The time measure will be took 20 times for each case, at the begin and the end of the Renderer Post-Update, with a Performance Timer made to measure performance in Milliseconds (ms), and then we will perform calculations with the average measures. This measures will be took in Release compile mode to test it in a more realistic environment.

Results

The final results are gathered in a excel sheet. Can be accessed here. To summarize, the first thing this teach us is to not to make any kind of assumption on performance expectatives. All three cases result to be pretty the same in terms of performance, even though, the first one is a bit (very bit) faster than the third, and the second a bit faster than the first, giving to the Case 2 the better place with an average of 0,52ms of rendering time. However, the differences are very small, the Case 1 lasts an average of 0,53ms and the 3rd Case, 0,55ms. Of course, for the 3rd Case is important to take into account the ordering time, but is very, very minimal, with an average of 0,00057 ms to order the whole vector, lasting 0,0032ms in the worst case measured-
If you enter the excel, you will see the resulting graphics of the time measurement for each case and a mix of the three graphics, and you will also see all the data in detail. We will let you those graphics also here. You will see that all cases are very similar in terms of performance.



Conclusions

To conclude, I think that this situation, a part of teaching us that is bad to circle around assumptions that we don't know if they are true or even if they will work (provoking us a lose of time), it also leave us in the hands of comfort. What I mean with this is that, since the differences between the cases are so small, we can choose the case that better fits to us. For us, in the context of a 3D Game Engine, will be the 2nd Case, not because is faster, but because it allows us to have more freedom in terms of working with many shaders and materials, and to support many shaders, we should keep binding/unbinding different shaders, and this is easier if we have the general data in an only place and pass it to all the shaders and, then, in each material, pass the material/object data to the shader it has attached.

Also, it's nice to mention at this stage that making this decision before would saved us from doing all this documentation and measures, and also, to invest time on doing a system to order objects as faster as possible, but there's a good side, always, which is that we have learned a bit more today!

You can visit Cronos Engine's web page and check also our repository. In there, you will find a branch with all what we have been seen in this document. Thanks for reading!

Annexes

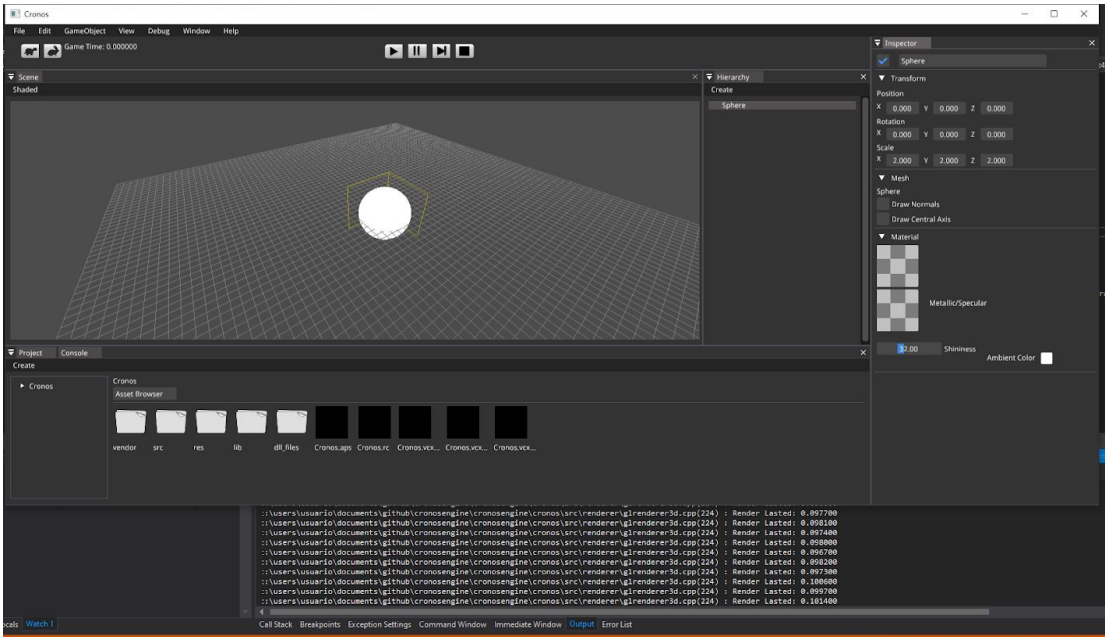
Objects Sorting Algorithm

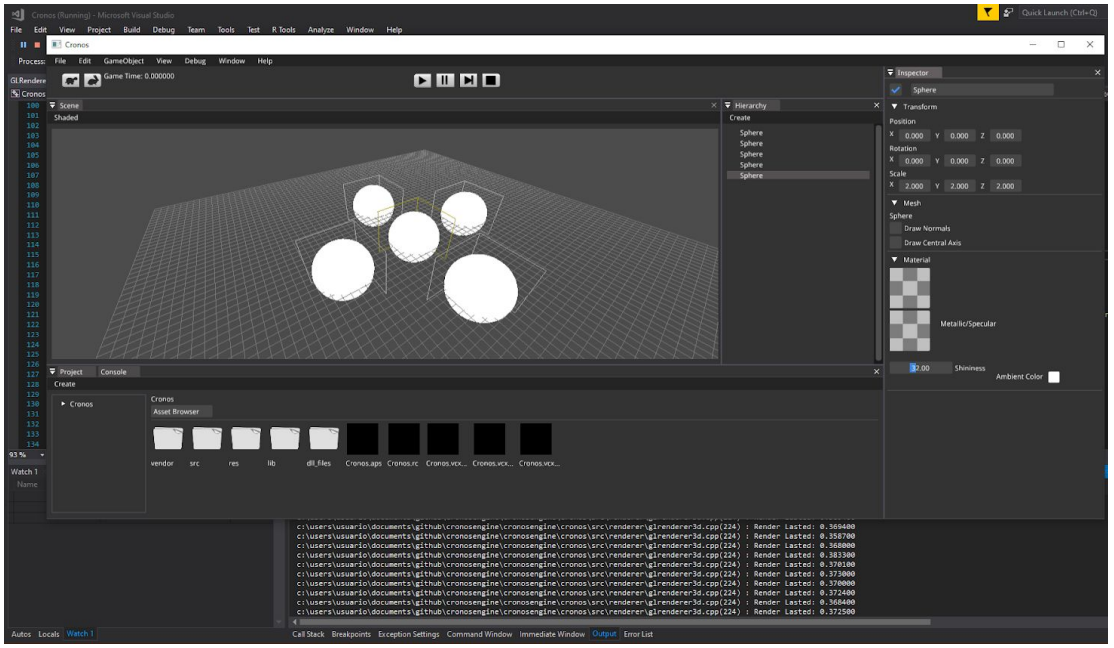
As you may have read, we needed a way of sorting the objects to render for the cases 3 and 4. Since this can influence this little investigation, It might be also important to explain here the algorithm used for that purpose, which is a very simple Quicksort algorithm that compares object's materials by their IDs and, recursively, puts them together inside the vector.

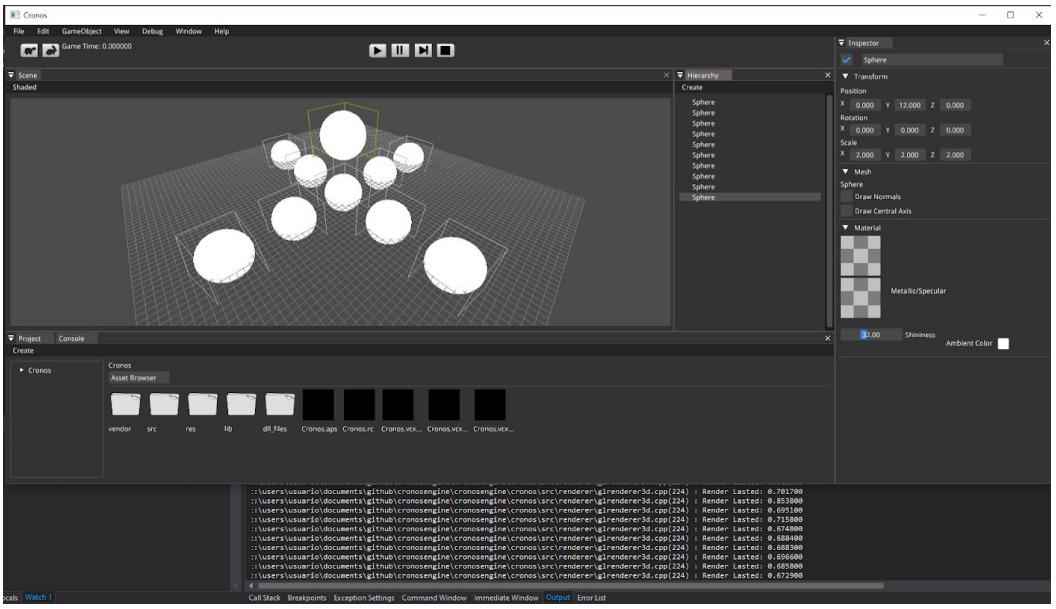
```
void GLRenderer3D::ObjectsSortByMaterial(std::vector<GameObject*>& objVec, uint left, uint right)
{
    if (left >= right || right >= objVec.size()) return;

    uint pivot_id = objVec[right]->GetComponent<MaterialComponent>()->GetMaterial()->GetMaterialID();
    uint cnt = left;

    for (uint i = left; i <= right; ++i)
    {
        if (objVec[i]->GetComponent<MaterialComponent>()->GetMaterial()->GetMaterialID() == pivot_id)
        {
            std::swap(objVec[cnt], objVec[i]);
            ++cnt;
        }
    }

    ObjectsSortByMaterial(objVec, left, cnt - 2);
    ObjectsSortByMaterial(objVec, cnt, right);
}
```

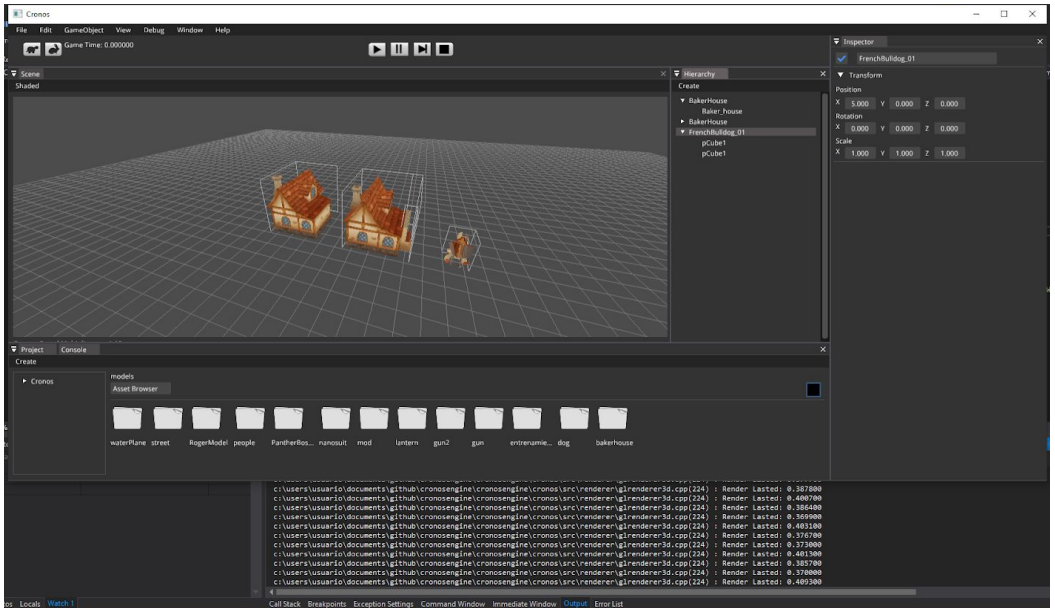
Tests & Measures Screenshots

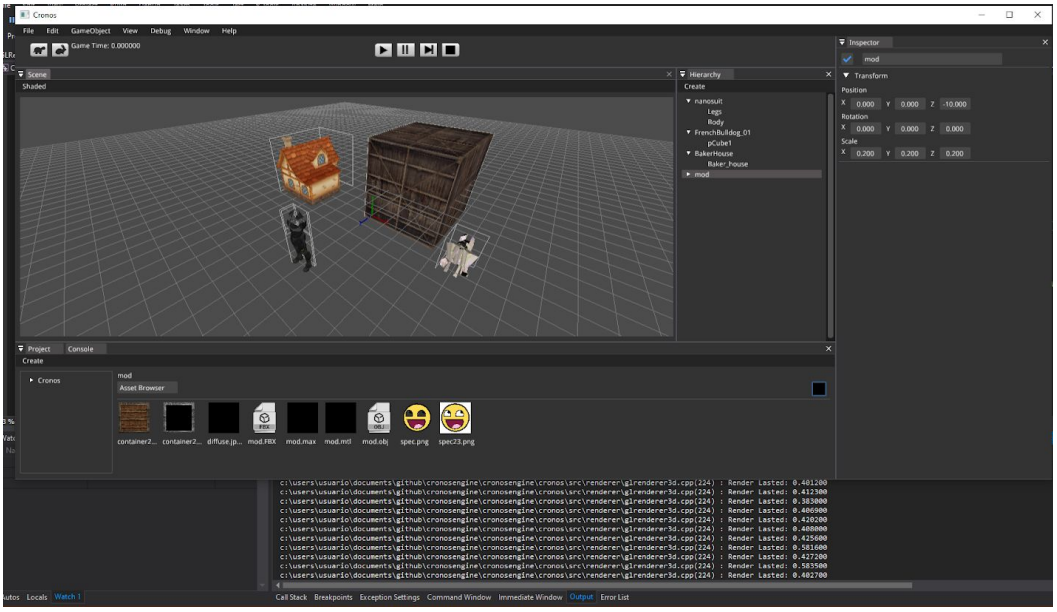
Case 1

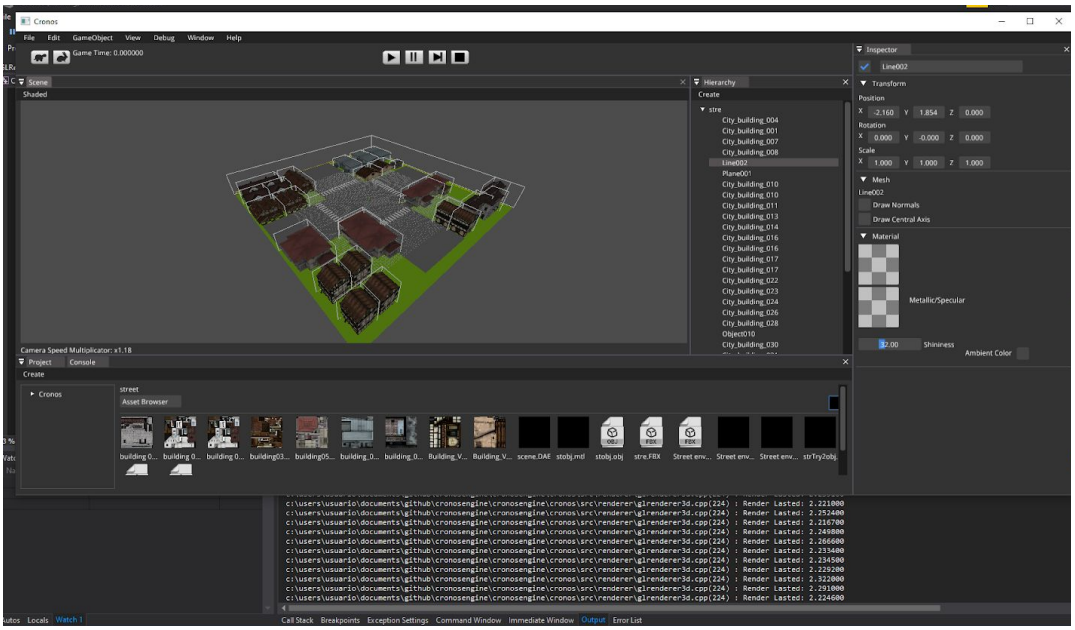


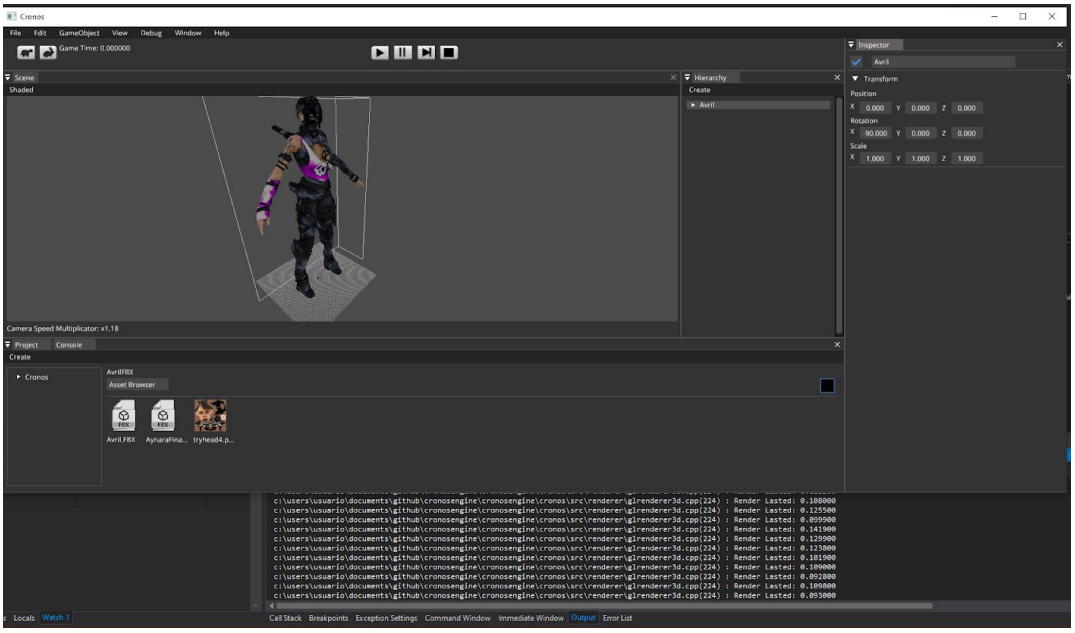



Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
Sphere
Inspector
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Project
Console
Asset Browser
vendor
src
res
lib
dll_files
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

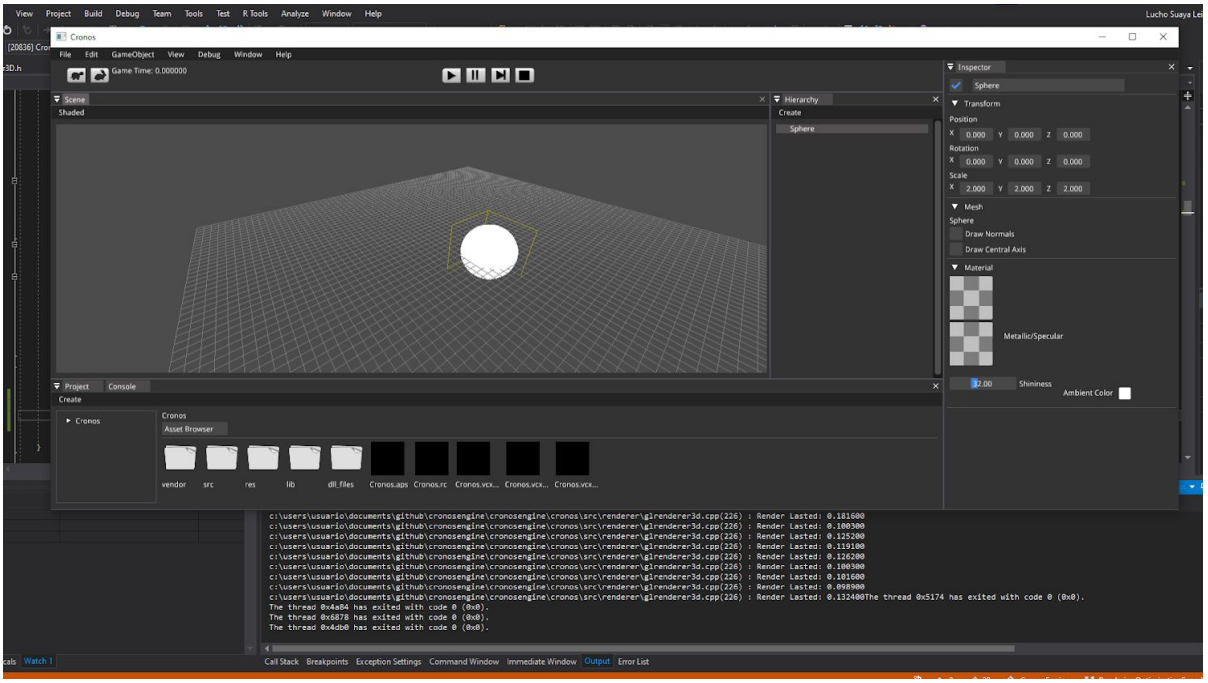

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
BakerHouse
Baker_house
FrenchBulldog_01
pCube1
Inspector
Transform
Position
Rotation
Scale
Project
Console
Asset Browser
models
waterPlane
street
RogerModel
people
PantherBos...
nanosuit
mod
luntern
gun2
gun
entrenamie...
dog
bakerhouse
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

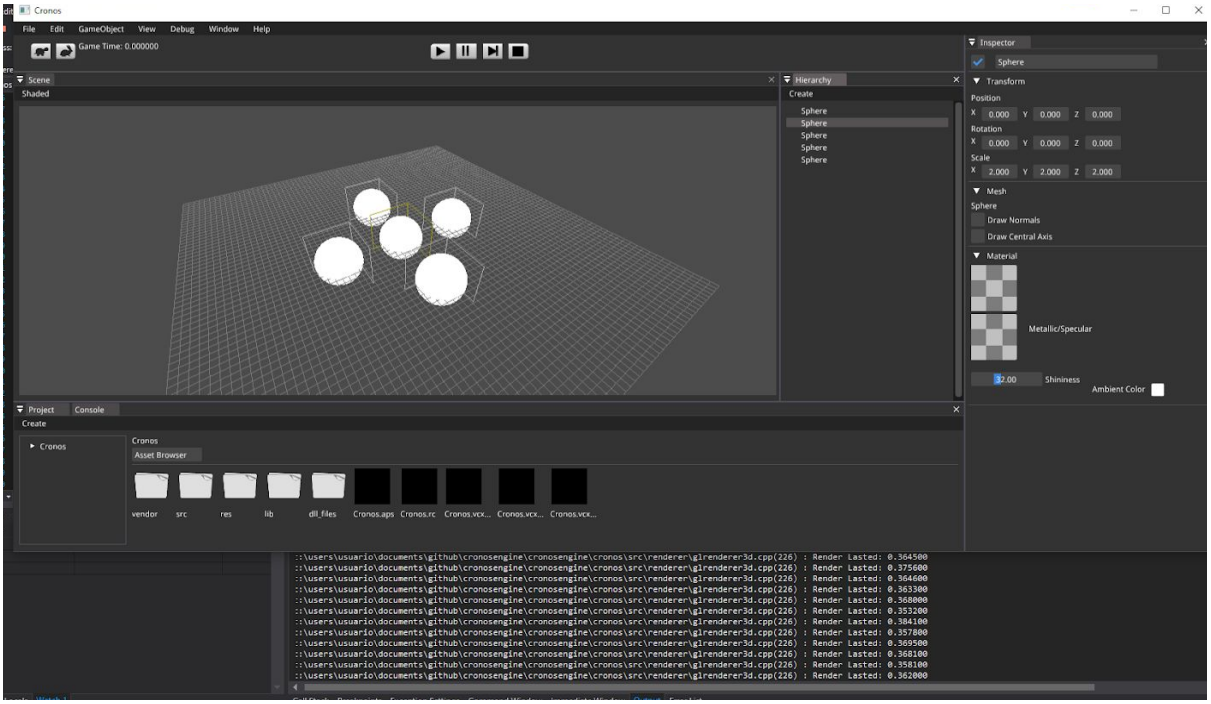

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
nanosuit
Legs
Body
FrenchBulldog_01
pCube1
BakerHouse
Baker_house
mod
Inspector
Transform
Position
Rotation
Scale
Project
Console
Asset Browser
container2...
container2...
diffuse.jp...
mod.FBX
mod.max
mod.mtl
mod.obj
spec.png
spec23.png
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

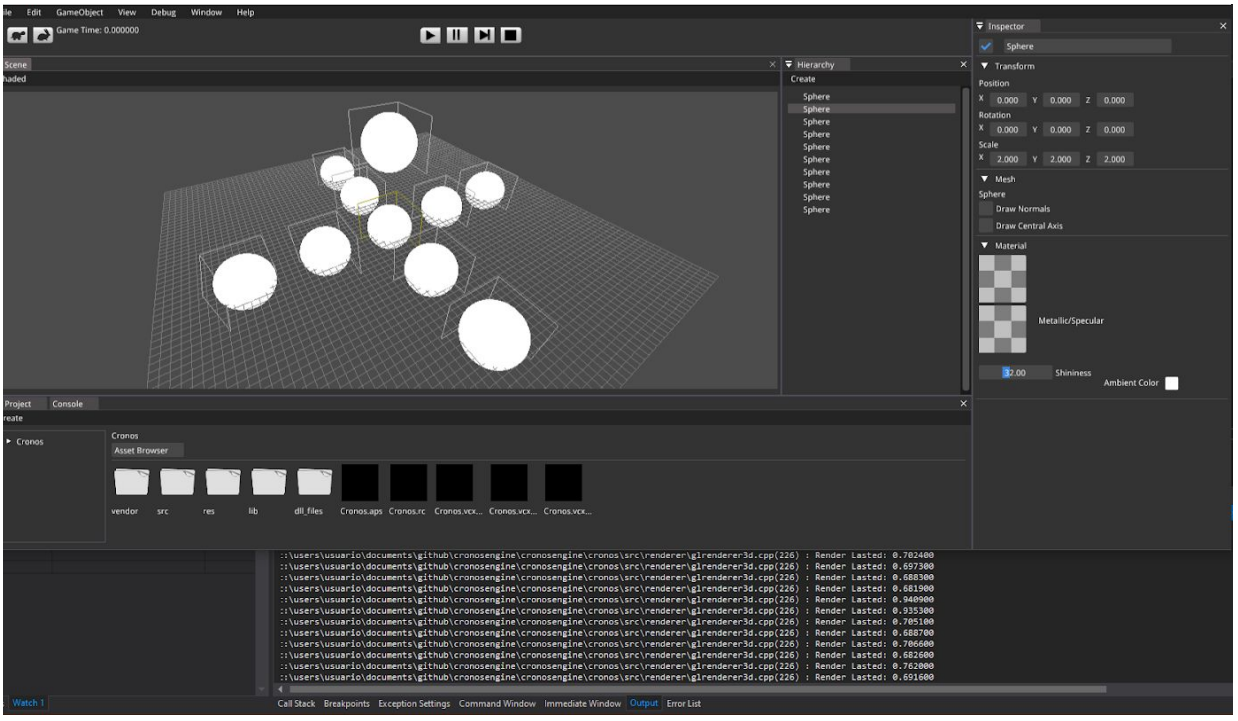

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
stre
City_building_004
City_building_001
City_building_007
City_building_008
Line002
Plane001
City_building_010
City_building_010
City_building_011
City_building_013
City_building_014
City_building_016
City_building_016
City_building_017
City_building_017
City_building_022
City_building_023
City_building_024
City_building_026
City_building_028
Object010
City_building_030
Inspector
Line002
Transform
Position
Rotation
Scale
Mesh
Line002
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Camera Speed Multiplicator: x1.18
Project
Console
Create
Cronos
street
Asset Browser
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

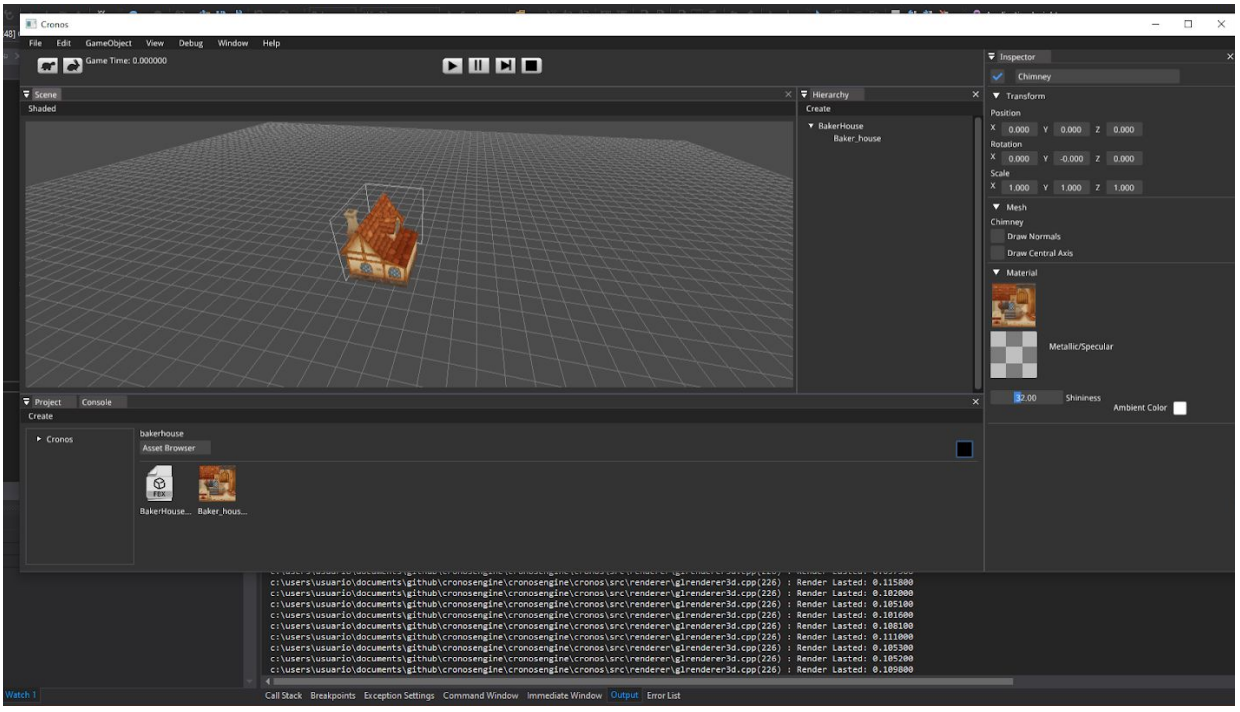

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
Avril
Inspector
Avril
Transform
Position
Rotation
Scale
Camera Speed Multiplicator: x1.18
Project
Console
Create
Cronos
AvrilFBX
Asset Browser
Avril.FBX
AynaraFina...
tryhead4.p...
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

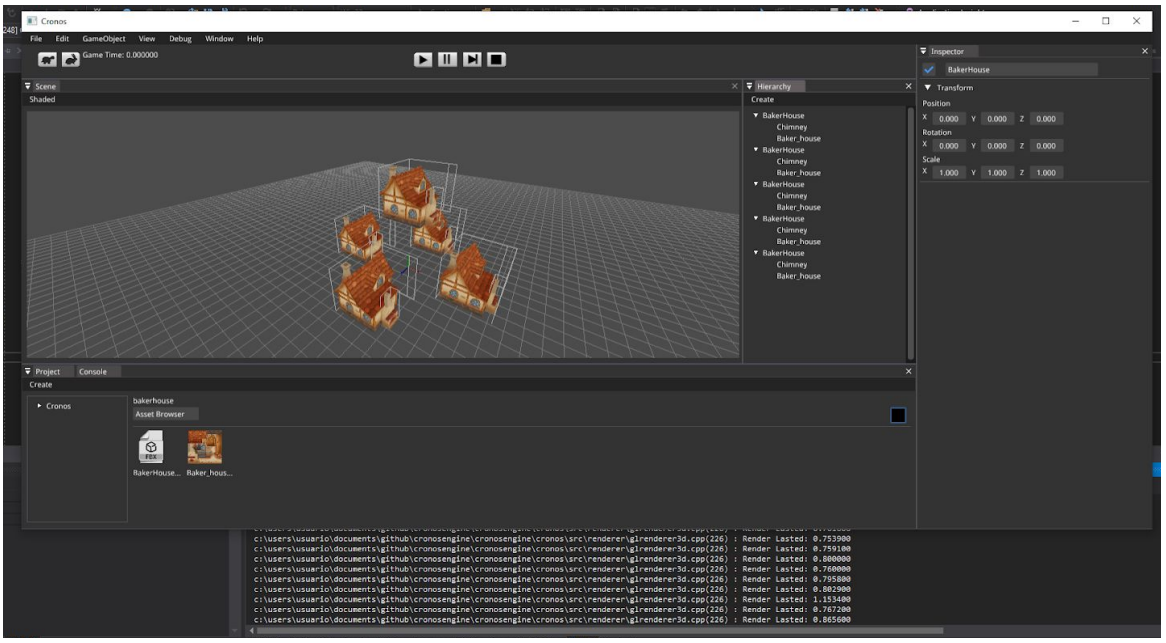
Case 2

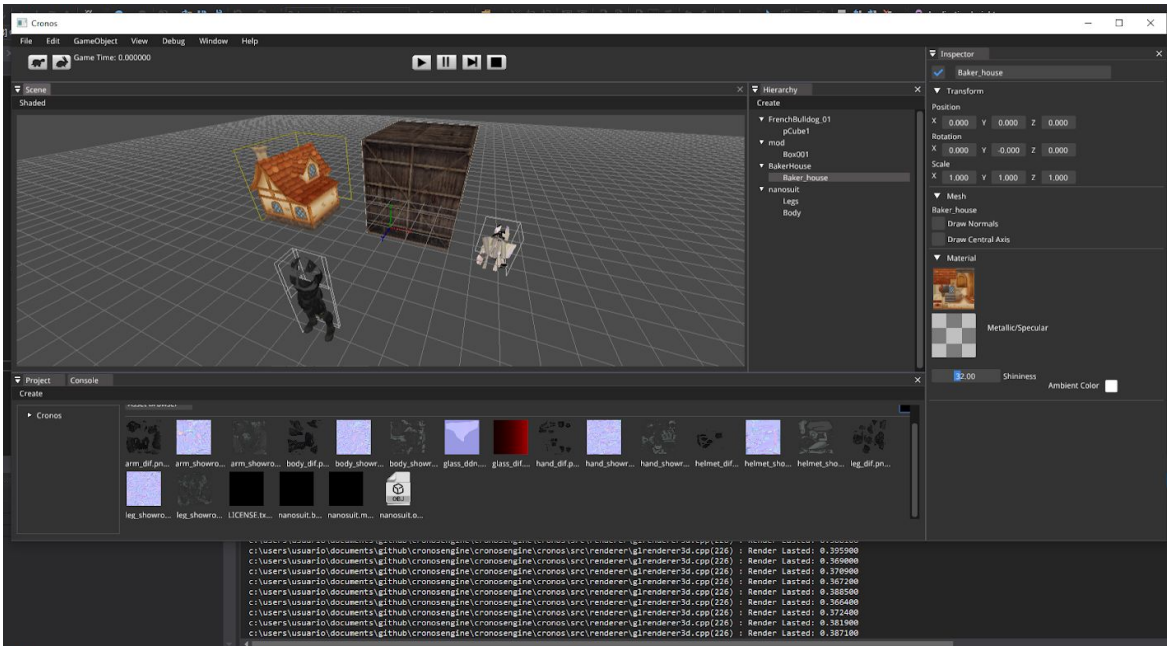

Cronos
Game Time: 0.000000
Inspector
Sphere
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Hierarchy
Create
Scene
Shaded
Project
Console
Asset Browser
vendor
src
res
lib
dll_files
Cronos.sln
Cronos.rc
Cronos.vcx...
Cronos.vcx...
Cronos.vcx...
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List


Cronos
Game Time: 0.000000
Inspector
Sphere
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Hierarchy
Create
Scene
Shaded
Project
Console
Asset Browser
vendor
src
res
lib
dll_files

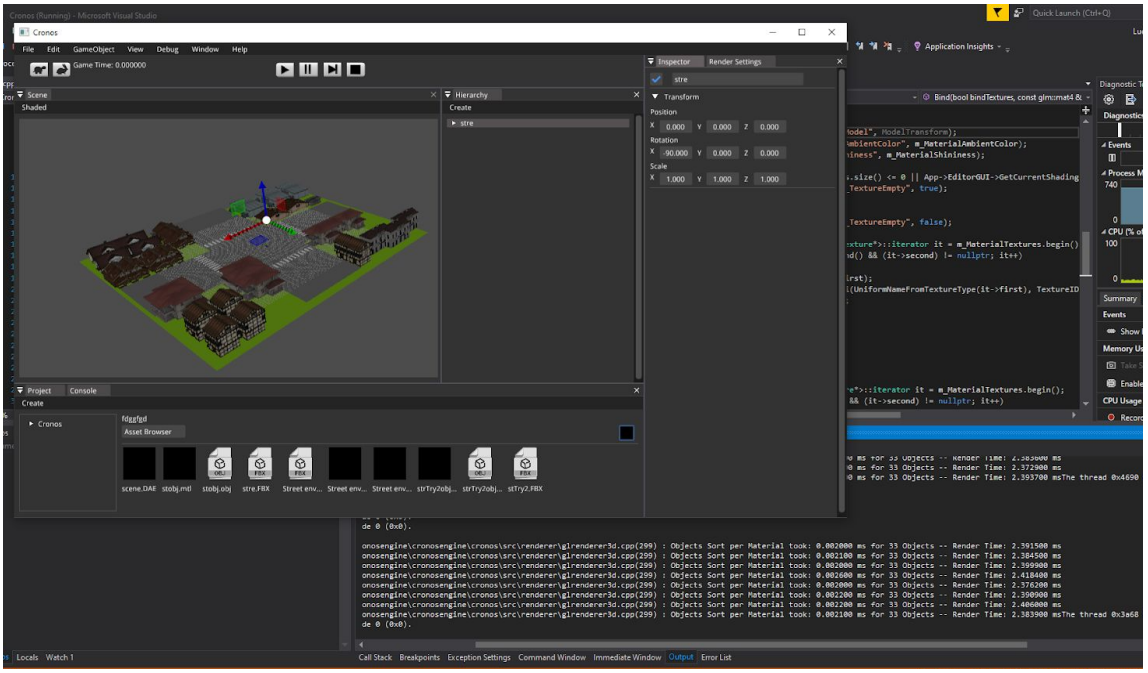

Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Hierarchy
Create
Sphere
Inspector
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Project
Console
Cronos
Asset Browser
vendor
src
res
lib
dll_files
Cronos.aps
Cronos.rc
Cronos.vcx...
Cronos.vcx...
Cronos.vcx...
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.702400
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.697300
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.688300
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.681900
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.940900
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.935300
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.705100
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.688700
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.706600
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.682600
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.762000
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.691600
Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

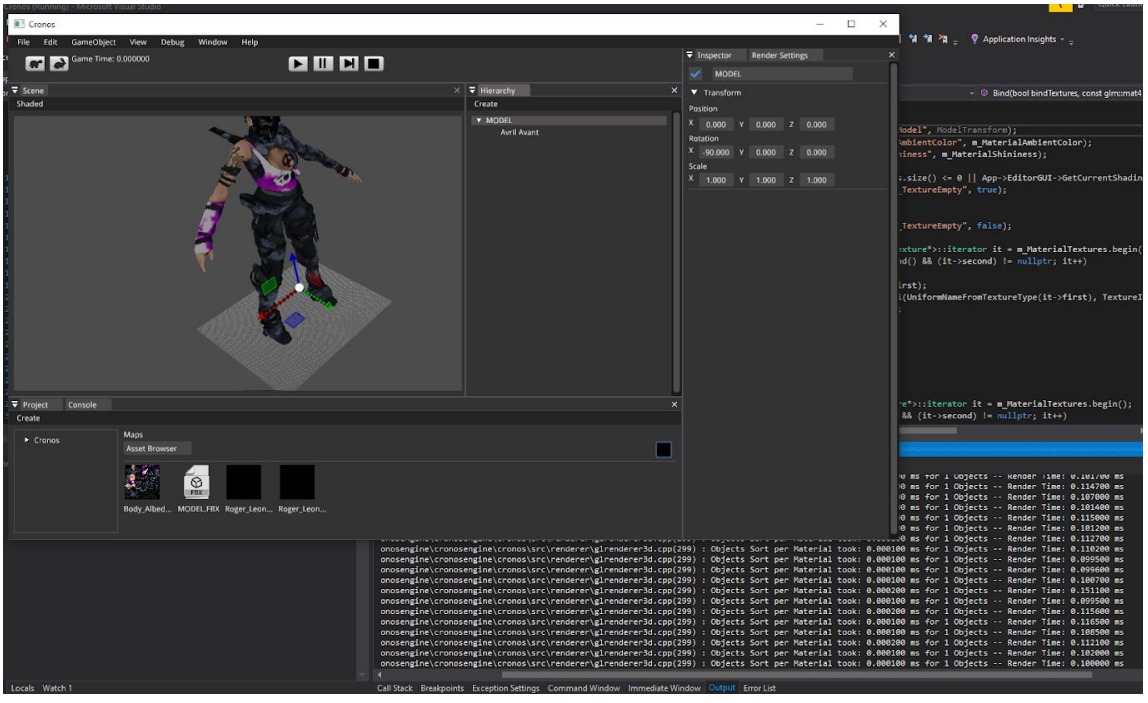

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
BakerHouse
Baker_house
Inspector
Chimney
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Project
Console
Cronos
bakerhouse
Asset Browser
BakerHouse...
Baker_hous...
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.115800
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.102000
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.105100
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.101600
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.108100
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.111000
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.105300
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.105200
::\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.109800
Watch 1
Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

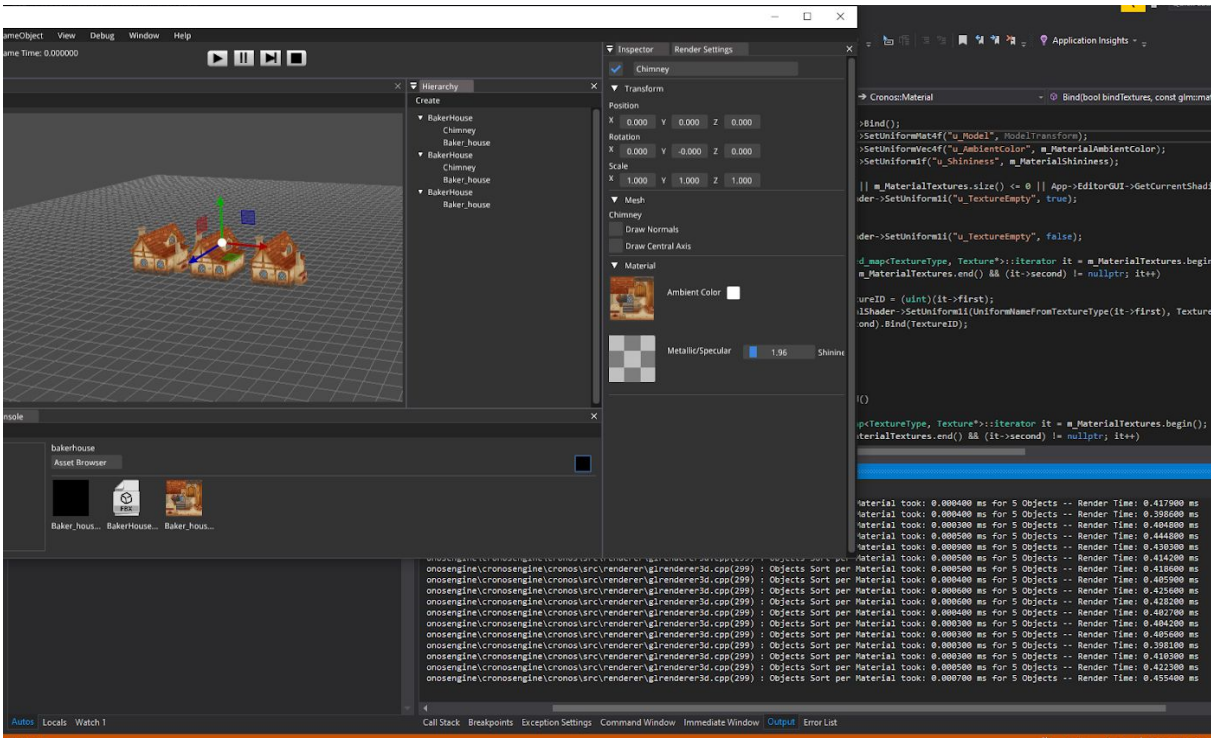

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
BakerHouse
Chimney
Baker_house
Inspector
BakerHouse
Transform
Position
Rotation
Scale
Project
Console
Create
Cronos
bakerhouse
Asset Browser
BakerHouse...
Baker_hous...
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.753900
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.759100
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.800000
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.760000
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.795800
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.802900
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 1.153400
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.767200
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.865600

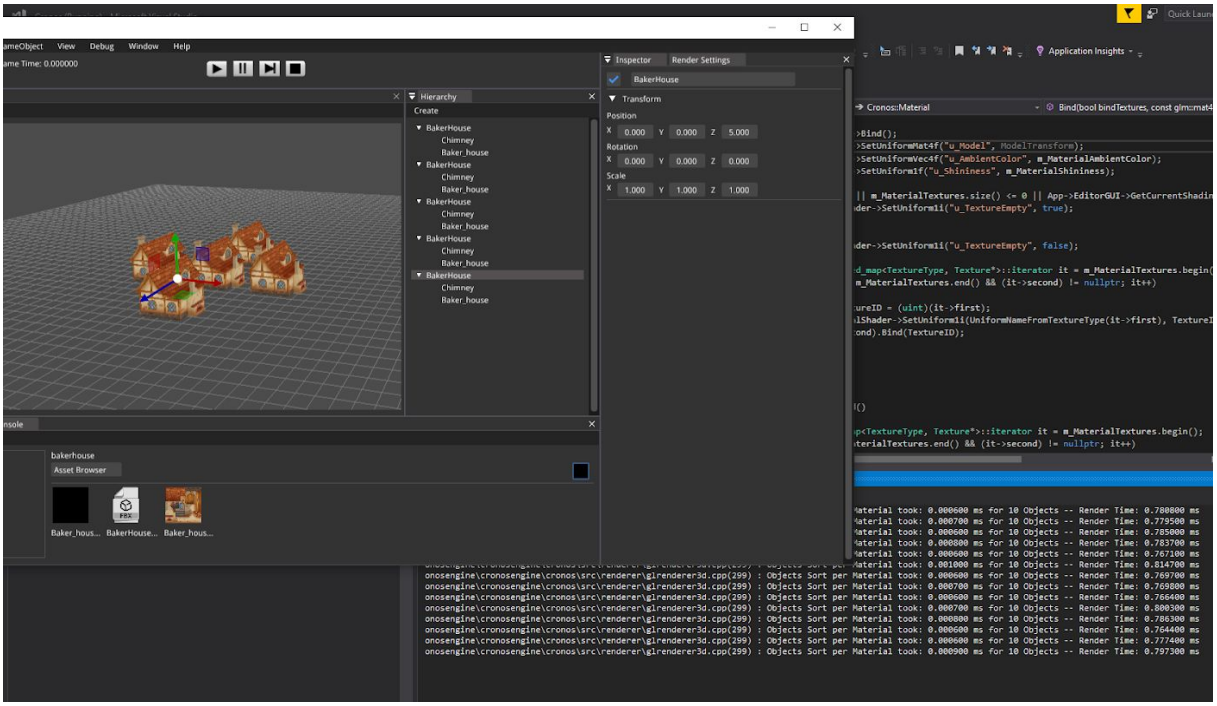

Cronos
File Edit GameObject View Debug Window Help
Game Time: 0.000000
Scene
Shaded
Hierarchy
Create
FrenchBulldog_01
pCube1
mod
Box001
BakerHouse
Baker_house
nanosuit
Legs
Body
Inspector
Baker_house
Transform
Position
Rotation
Scale
Mesh
Baker_house
Draw Normals
Draw Central Axis
Material
Metallic/Specular
Shininess
Ambient Color
Project
Console
Create
Cronos
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.395900
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.369000
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.370900
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.367200
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.388500
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.366400
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.372400
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.381900
c:\users\usuario\documents\github\cronosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(226) : Render Lasted: 0.387100

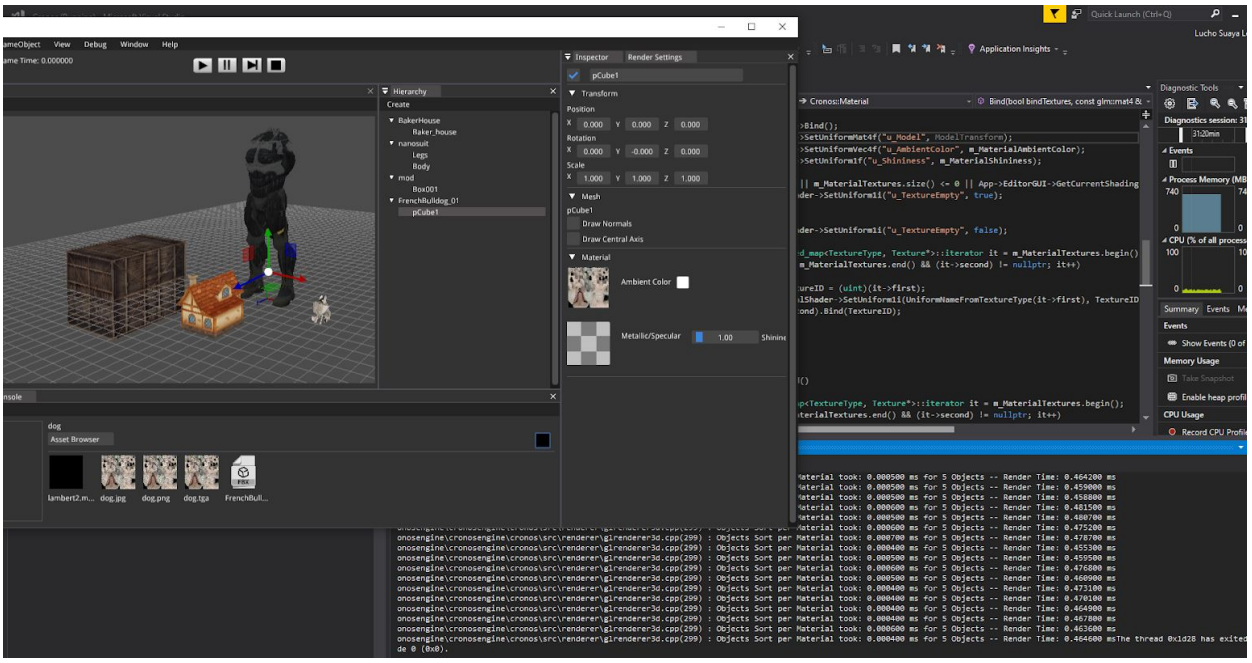
Case 3

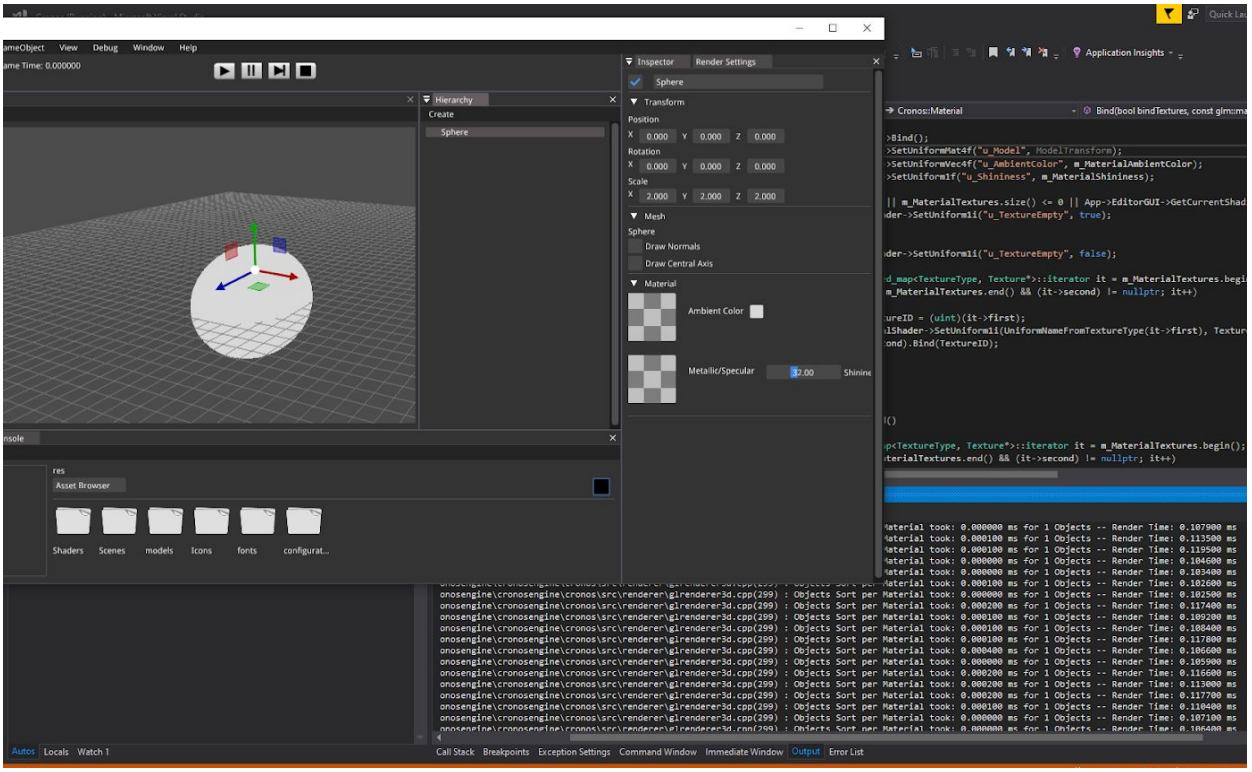


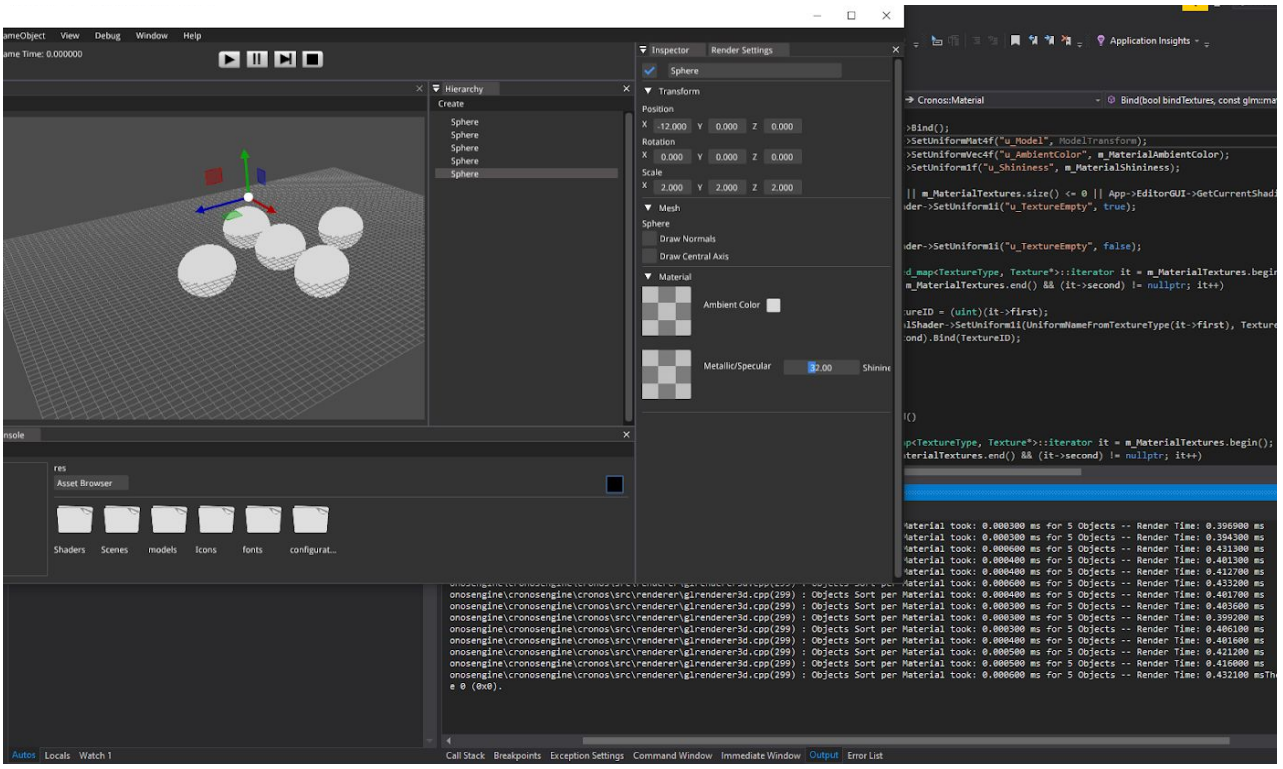


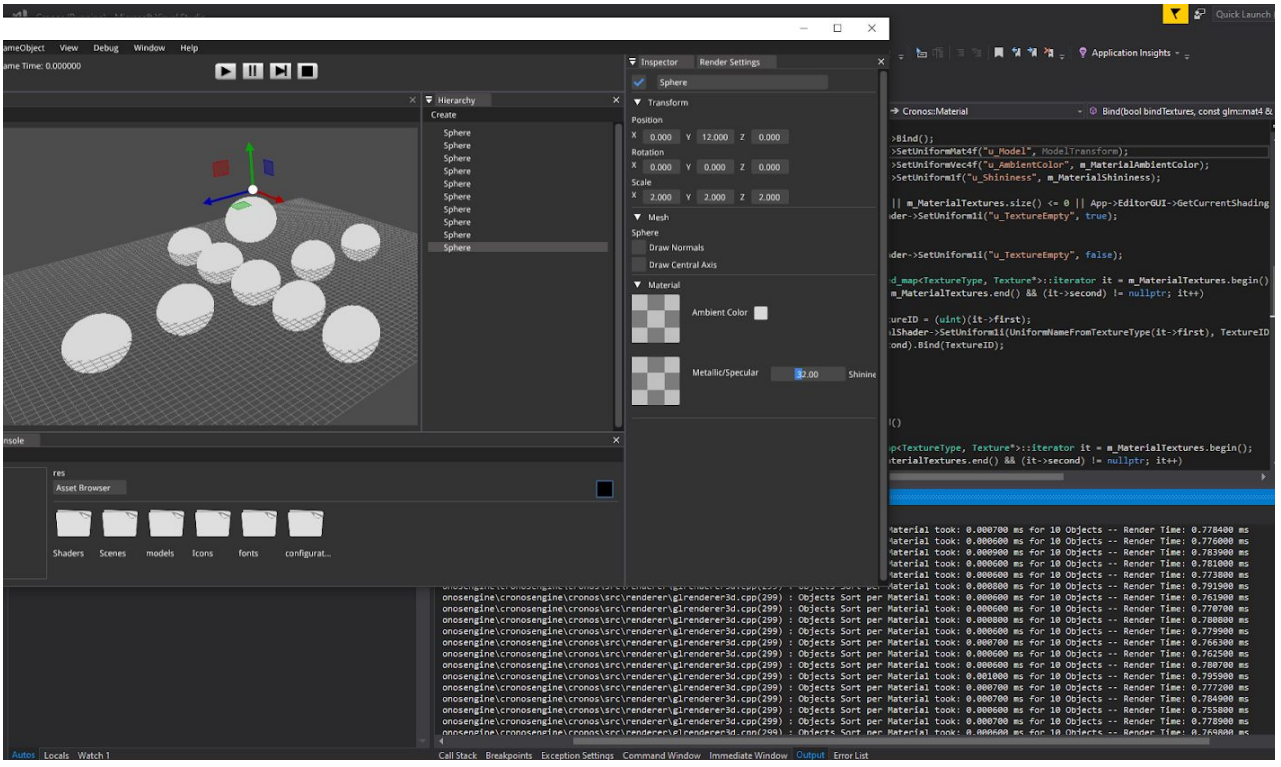

Hierarchy
Create
BakerHouse
Chimney
Baker_house
Inspector
Render Settings
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Ambient Color
Metallic/Specular
Asset Browser
Application Insights
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List

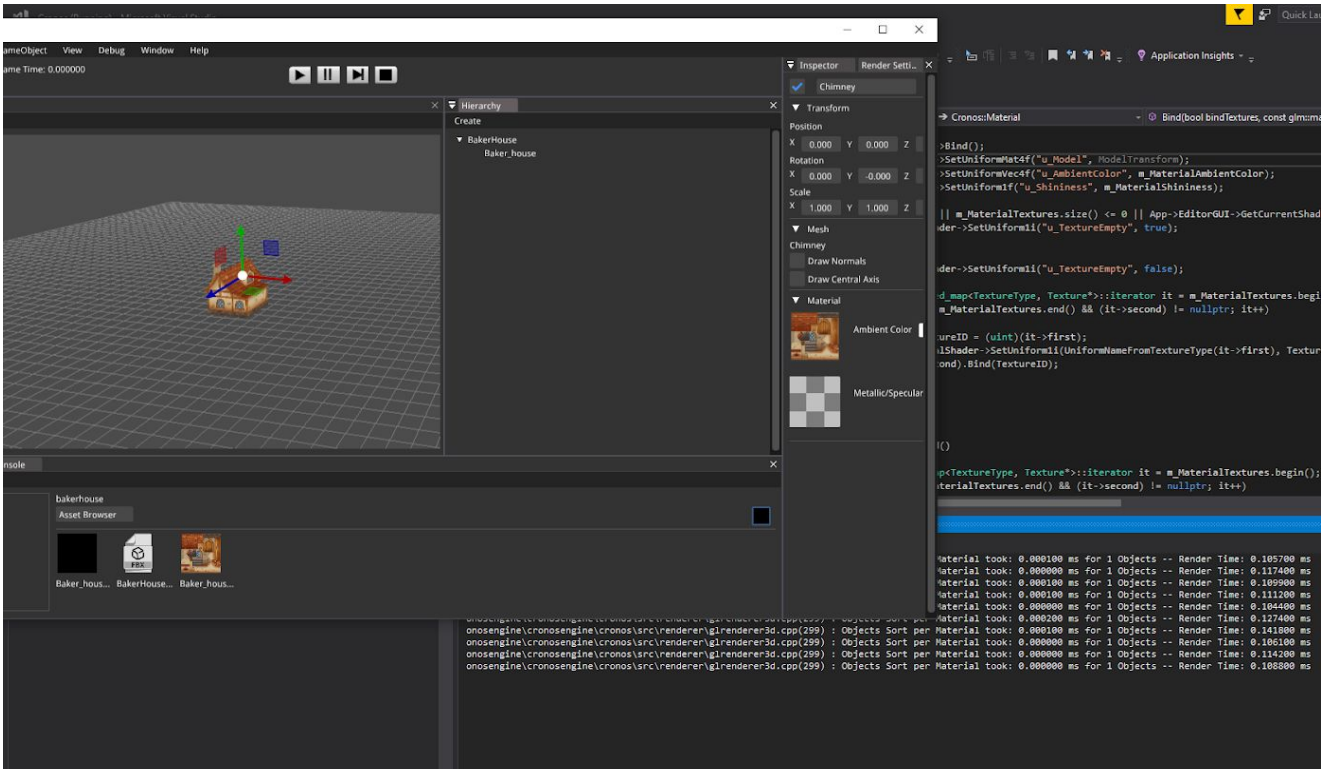

Hierarchy
Create
BakerHouse
Chimney
Baker_house
Inspector
Render Settings
Transform
Position
Rotation
Scale
Asset Browser
Application Insights






Inspector
Render Settings
Sphere
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Ambient Color
Metallic/Specular
Hierarchy
Create
Asset Browser
Shaders
Scenes
models
Icons
fonts
Application Insights
Cronos::Material
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List


Inspector
Render Settings
Sphere
Transform
Position
Rotation
Scale
Mesh
Draw Normals
Draw Central Axis
Material
Ambient Color
Metallic/Specular
Hierarchy
Create
Asset Browser
Shaders
Scenes
models
Icons
fonts
Application Insights
Cronos::Material
Call Stack
Breakpoints
Exception Settings
Command Window
Immediate Window
Output
Error List


ameObject View Debug Window Help
ame Time: 0.000000
Hierarchy
Create
BakerHouse
Baker_house
Inspector
Render Setti...
Chimney
Transform
Position
Rotation
Scale
Mesh
Chimney
Draw Normals
Draw Central Axis
Material
Ambient Color
Metallic/Specular
nsole
bakerhouse
Asset Browser
Baker_hous...
BakerHouse...
Baker_hous...
Application Insights
Cronos::Material
Bind(bool bindTextures, const glm::ma
>Bind();
>SetUniformMat4f("u_Model", ModelTransform);
>SetUniformVec4f("u_AmbientColor", m_MaterialAmbientColor);
>SetUniform1f("u_Shininess", m_MaterialShininess);
|| m_MaterialTextures.size() <= 0 || App->EditorGUI->GetCurrentShad
der->SetUniform1i("u_TextureEmpty", true);
der->SetUniform1i("u_TextureEmpty", false);
d_map<TextureType, Texture*>::iterator it = m_MaterialTextures.begi
m_MaterialTextures.end() && (it->second) != nullptr; it++)
ureID = (uint)(it->first);
lShader->SetUniform1i(UniformNameFromTextureType(it->first), Textur
ond).Bind(TextureID);
l()
p<TextureType, Texture*>::iterator it = m_MaterialTextures.begin();
terialTextures.end() && (it->second) != nullptr; it++)
Material took: 0.000100 ms for 1 Objects -- Render Time: 0.105700 ms
Material took: 0.000000 ms for 1 Objects -- Render Time: 0.117400 ms
Material took: 0.000100 ms for 1 Objects -- Render Time: 0.109900 ms
Material took: 0.000100 ms for 1 Objects -- Render Time: 0.111200 ms
Material took: 0.000000 ms for 1 Objects -- Render Time: 0.104400 ms
Material took: 0.000200 ms for 1 Objects -- Render Time: 0.127400 ms
onosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(299) : Objects Sort per Material took: 0.000100 ms for 1 Objects -- Render Time: 0.141800 ms
onosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(299) : Objects Sort per Material took: 0.000000 ms for 1 Objects -- Render Time: 0.106100 ms
onosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(299) : Objects Sort per Material took: 0.000000 ms for 1 Objects -- Render Time: 0.114200 ms
onosengine\cronosengine\cronos\src\renderer\glrenderer3d.cpp(299) : Objects Sort per Material took: 0.000000 ms for 1 Objects -- Render Time: 0.108800 ms