FIU School of Computing & Information Sciences

Senior Project, 2013, Fall

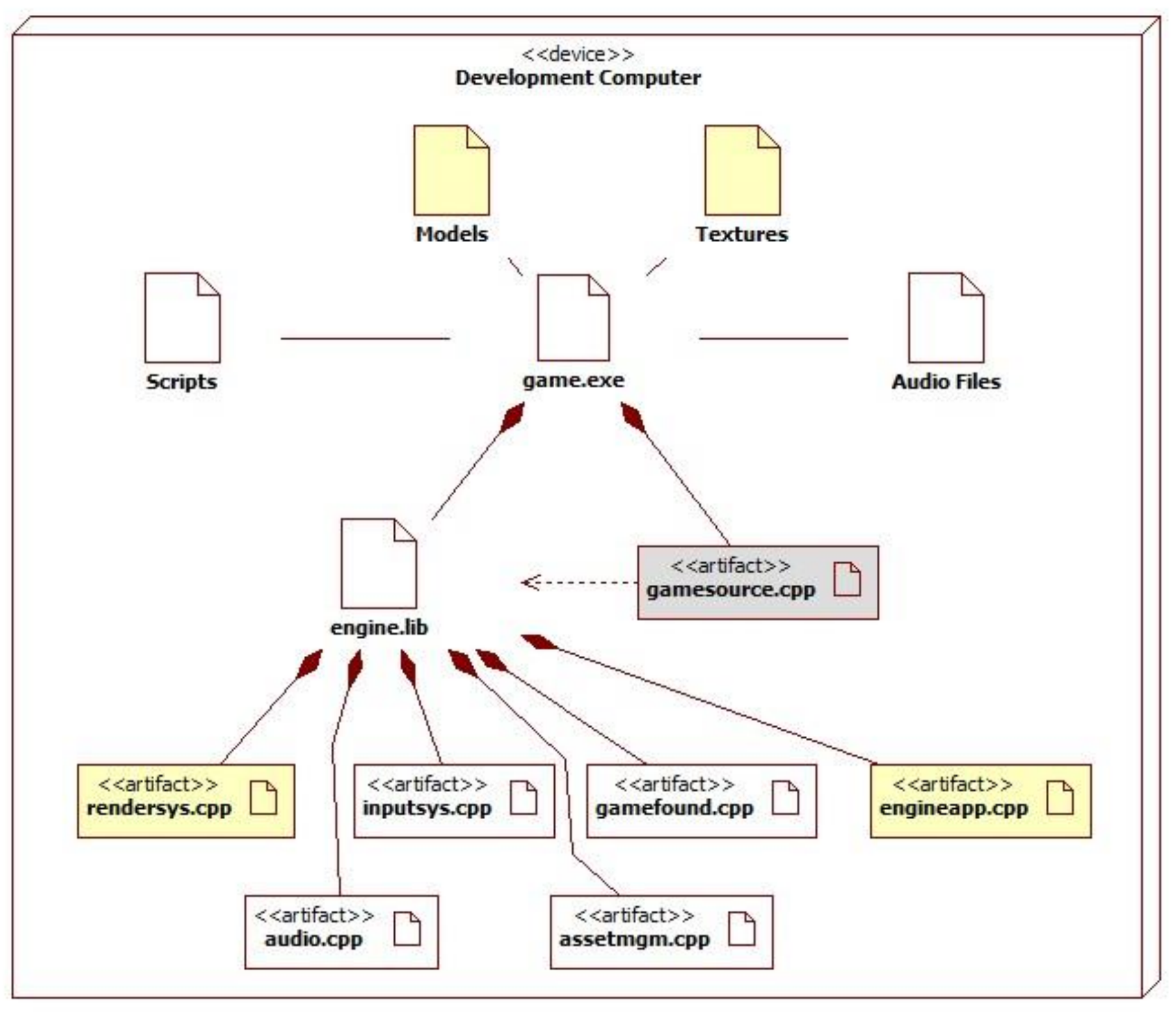
GAME ENGINE

Student: Michael Weschler, Florida International University
Mentor: *Wei Zeng, Florida International University*
Instructor: Masoud Sadjadi, Florida International University

PROBLEM

- Most modern open source game engines do not provide an easy to use structure for novice programmers to build their game on
- In addition to an easy to use framework, an easy yet powerful rendering system is also needed. Many novice friendly systems do not allow access to the more powerful shaders that fuel modern games.
- Rendering pipelines are often difficult to start and maintain at the low level, especially for novice programmers

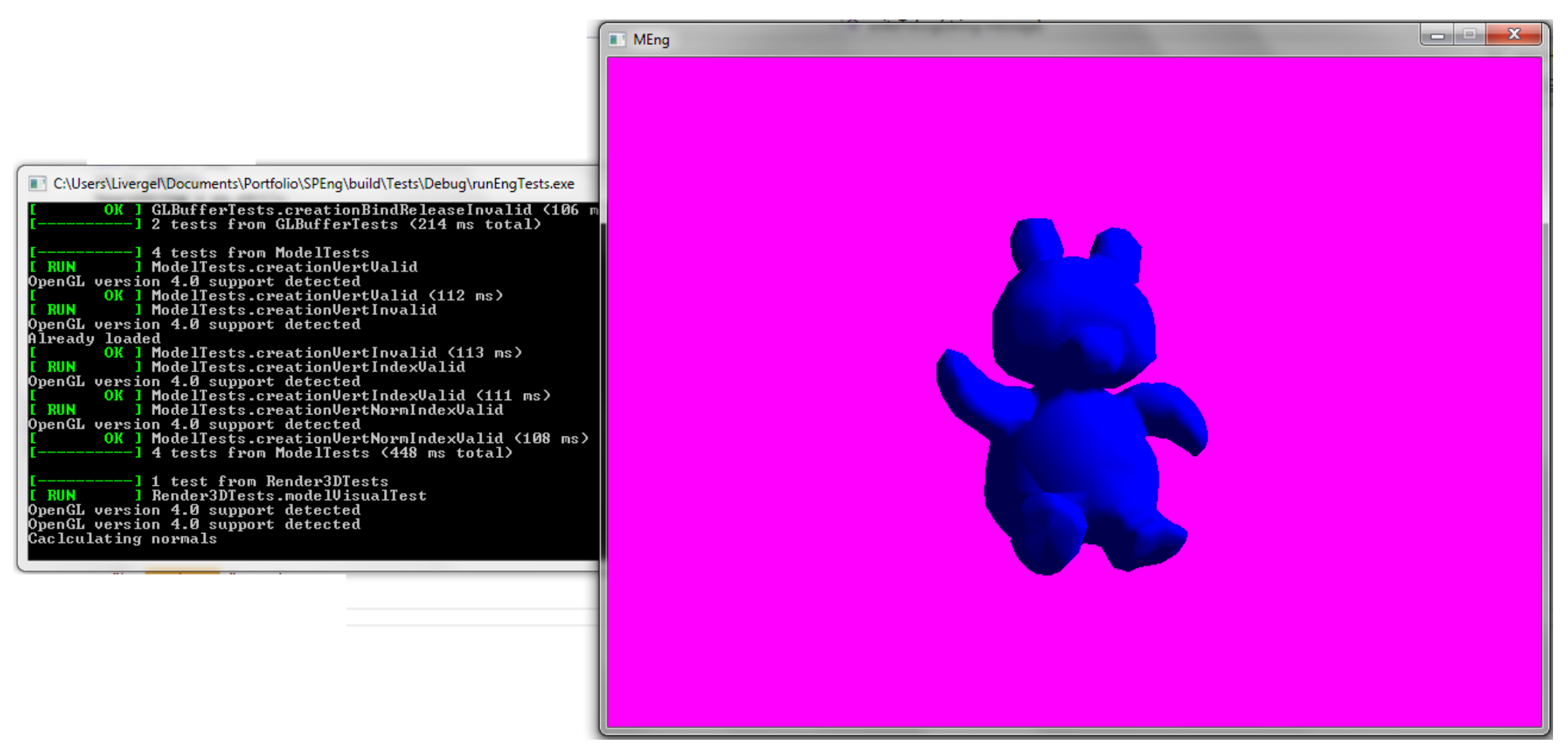
SYSTEM DESIGN


<<device>>
Development Computer
Models
Textures
Scripts
game.exe
Audio Files
engine.lib
<<artifact>> gamesource.cpp
<<artifact>> rendersys.cpp
<<artifact>> inputsys.cpp
<<artifact>> gamefound.cpp
<<artifact>> engineapp.cpp
<<artifact>> audio.cpp
<<artifact>> assetmgm.cpp

The engine is designed to be a series of layers that, when compiled with the user's code and assets, makes a complete game. My systems, in yellow, along with the other systems are compiled into a library. Along with he user's code, in grey, will be linked together to form the final game executable.

VERIFICATION

For testing, Googletest was used. This framework provides a suite of tools for unit tests in C++. The project as a whole was integrated using a bottom up approach which worked naturally with the layered design. Some of the rendering required manual visual inspection of the results as OpenGL does not provide a convenient way to verify rendering results.



```
C:\Users\Livergel\Documents\Portfolio\SPEng\build\Tests\Debug\runEngTests.exe
[       OK ] GLBufferTests.creationBindReleaseInvalid (106 m
[----------] 2 tests from GLBufferTests (214 ms total)

[----------] 4 tests from ModelTests
[ RUN      ] ModelTests.creationVertValid
OpenGL version 4.0 support detected
[       OK ] ModelTests.creationVertValid (112 ms)
[ RUN      ] ModelTests.creationVertInvalid
OpenGL version 4.0 support detected
Already loaded
[       OK ] ModelTests.creationVertInvalid (113 ms)
[ RUN      ] ModelTests.creationVertIndexValid
OpenGL version 4.0 support detected
[       OK ] ModelTests.creationVertIndexValid (111 ms)
[ RUN      ] ModelTests.creationVertNormIndexValid
OpenGL version 4.0 support detected
[       OK ] ModelTests.creationVertNormIndexValid (108 ms)
[----------] 4 tests from ModelTests (448 ms total)

[----------] 1 test from Render3DTests
[ RUN      ] Render3DTests.modelVisualTest
OpenGL version 4.0 support detected
OpenGL version 4.0 support detected
Caclculating normals
```

Unit tests and a visual test

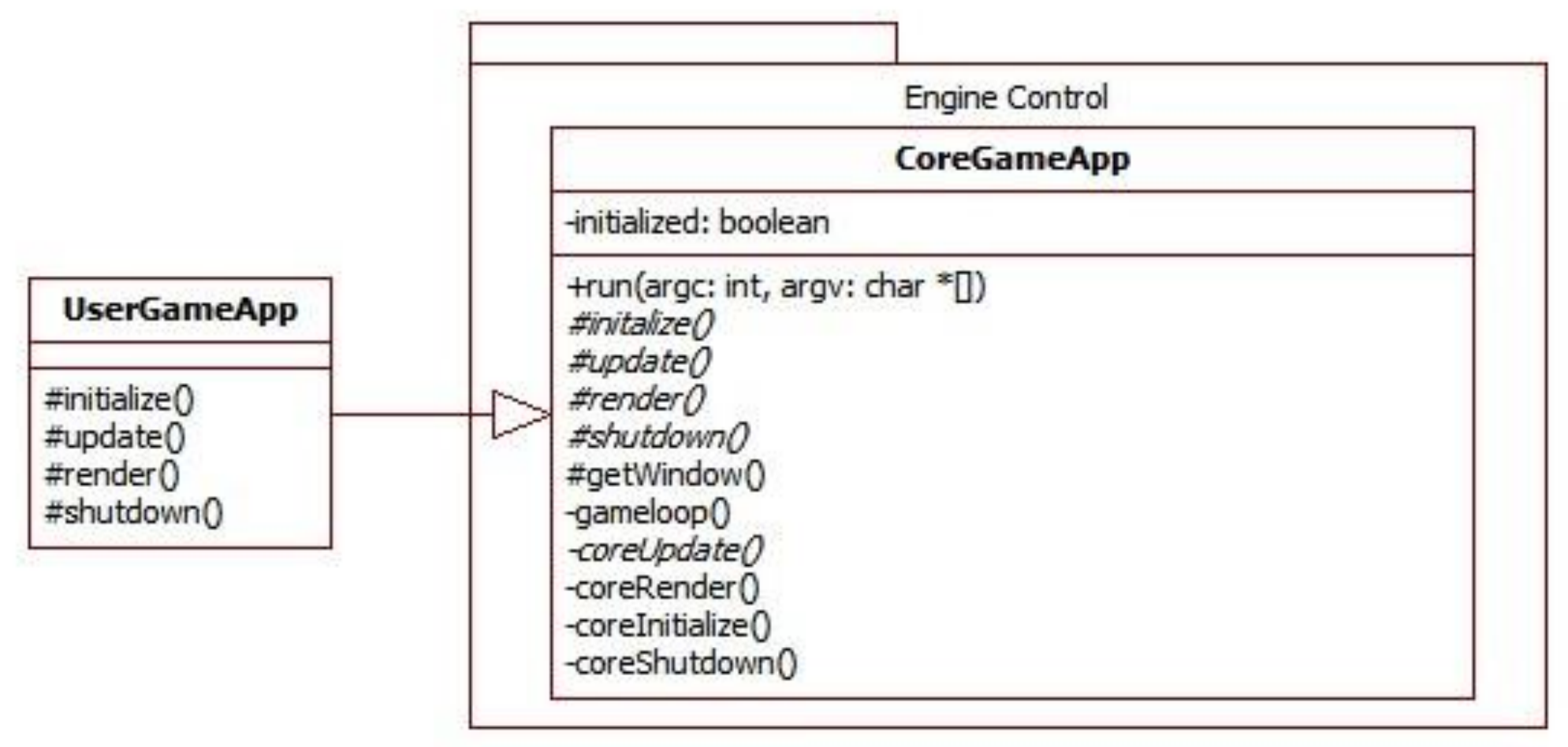
CURRENT SYSTEM

Ogre 3D is an open source rendering engine. While it provides extensive rendering functionality, It does not provide many of the other features for game design.

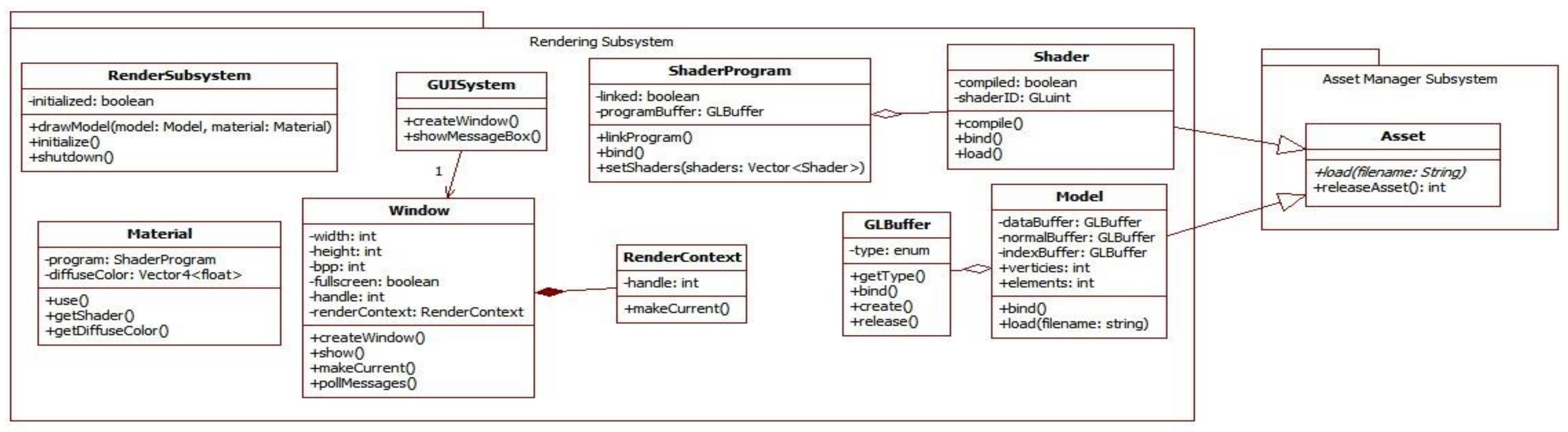
Moai provides and open source game engine that allows the user to create the game with Lua. Unfortunately, this is the only way a user may create a game. By forcing the user to only use Lua paired with the unique architecture makes it impractical for larger scale games.

The Unreal Engine is a very popular game engine used throughout the industry. However, its very high cost typically make it an unreasonable choice for smaller studios.

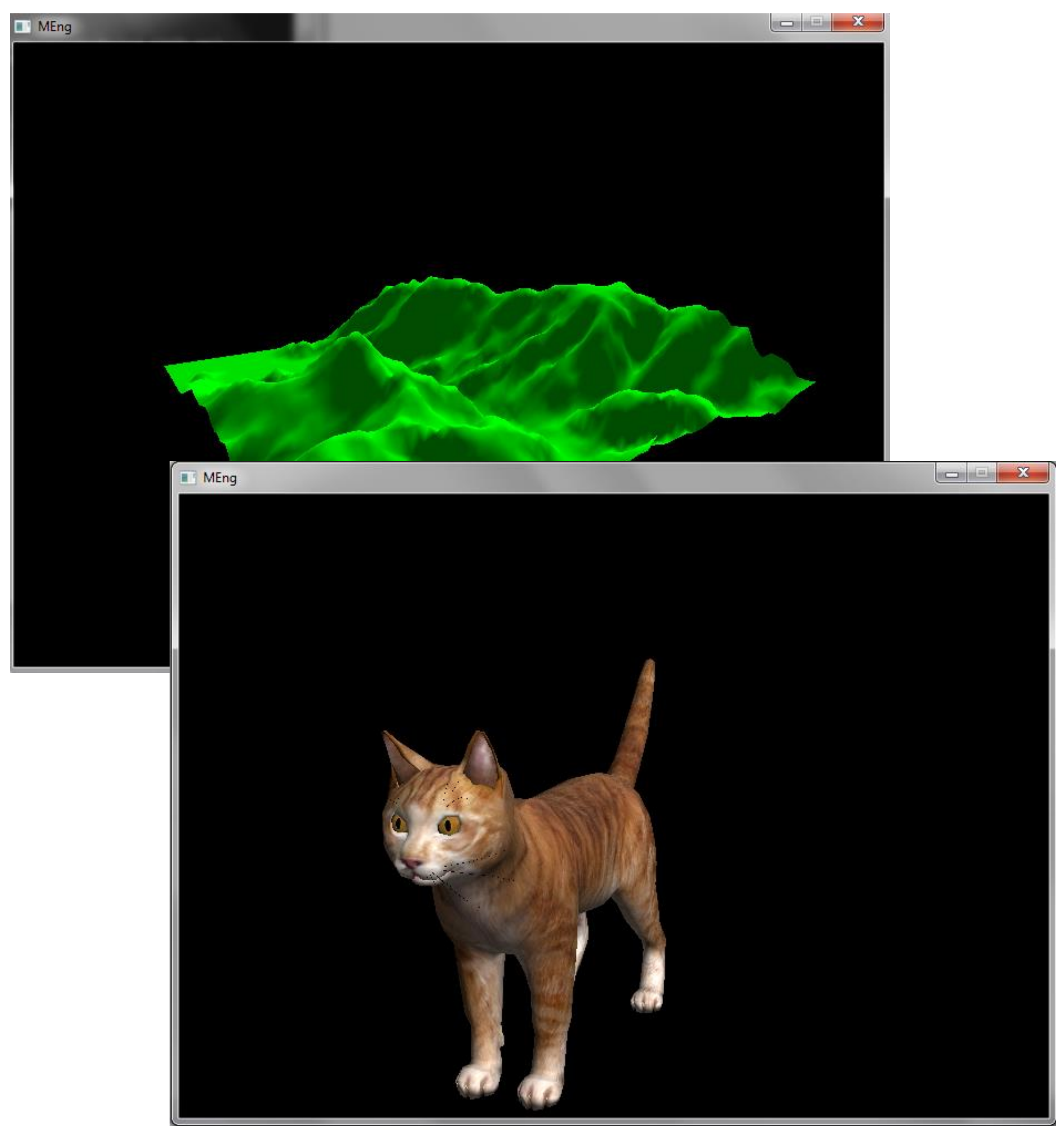
OBJECT DESIGN



Above: A core application class is inherited from by the user to have a framework for their game.
Below: The rendering subsystem is comprised of several container classes. User interfaces with this subsystem through a façade.



SCREENSHOTS



Diffuse colored model and textured model rendered in real time

REQUIREMENTS

The System Shall…
•Provide the ability to render 2D/3D graphics, these graphics will be represented by assets like models textures, and shaders.

•Manage and maintain a rendering pipeline for the user.

•Provide a way to initialize and control the application using a core game loop that will be maintained by the user.

•Provide an structure for the user to build their game within.

IMPLEMENTATION

The engine was written in C/C++ with the rendering system interfacing with OpenGL. The rendering system uses shaders, programs written in GLSL and compiled on the graphics hardware, to achieve its rendering. Much of the work of communicating with OpenGL is abstracted away through a series of representative classes.

The user can use the engine through a CoreGameApp class. This class can be inherited from to create a custom application class that gives the user a framework for their game. All the user has to do is run the application and it will initialize core and users subsystems, run the game loop, and then shutdown when told to quit.

SUMMARY

There are not many easy to use yet powerful game engines available to novice programmers. Even less are available that are also open source. The game engine we designed fills that space. It provides a broad set of features and an easy to use framework that is open to customization by the user.

The framework allows a programmer to create a game in a combination of C++ and Lua to fit their skill level. This gives them a fill in the blanks, yet flexible, project.

The rendering system also provides a simple yet powerful system to easily draw 2D and 3D assets to the screen. Through shaders they can achieve nearly any effect and look they desire.

ACKNOWLEDGEMENT

The material presented in this poster is based upon the work by Antonio Diaz, Brian Lara, Robert Law, and Julian Nodarse. Thanks to the whole group for their hard work. Additional thanks to Wei Zeng and Peter Clark for their feedback and support.