Engine Scripting with Lua

Lion Kortlepel `@lionkor`

July 12, 2020

Contents

Abstract

This engine lets you script behavior in Lua. In order for this to be useful, the engine provides a number of functions and globals, which are documented in this PDF. Further, it will show how to use Lua to script some basic behavior by providing some examples.

1 Setup

For scripts to be read by the engine, an `Entity` needs to have a `ScriptableComponent`. The constructor of the latter accepts a scriptfile name, like `example.lua`. For the file to be loaded it needs to be in the `Data/` directory and listed in the `Data/res.list` of your project.

An example program follows that will be used for the rest of this pdf.

```cpp
#include "Engine.h"
int main() {
    // Create an application
    Application app("Scripting101 Program", { 800, 600 });
    // Create an entity
    WeakPtr<Entity> entity_weak = app.add_entity();
    // Lock the Ptr for temporary access
    auto entity = entity_weak.lock();
    // Add ScriptableComponent
    entity->add_component<ScriptableComponent>("example.lua");
    // Run the application
    return app.run();
}
```

example.cpp

Additionally, the files `Data/example.lua` and `Data/res.list` exist.

```lua
Engine.log_info("Hello, World!")
```

Data/example.lua

```
example.lua
```

Data/res.list

With these files in place and the `example.cpp` compiled, we can now write code in `Data/example.lua`.

2 Exposed Functions

2.1 Engine API

The `Engine` namespace provides general engine functionality, mostly used for debugging and core engine functionalities.

2.1.1 Engine.log_info(message)

```lua
Engine.log_info("Hello, World!")
```

Data/example.lua