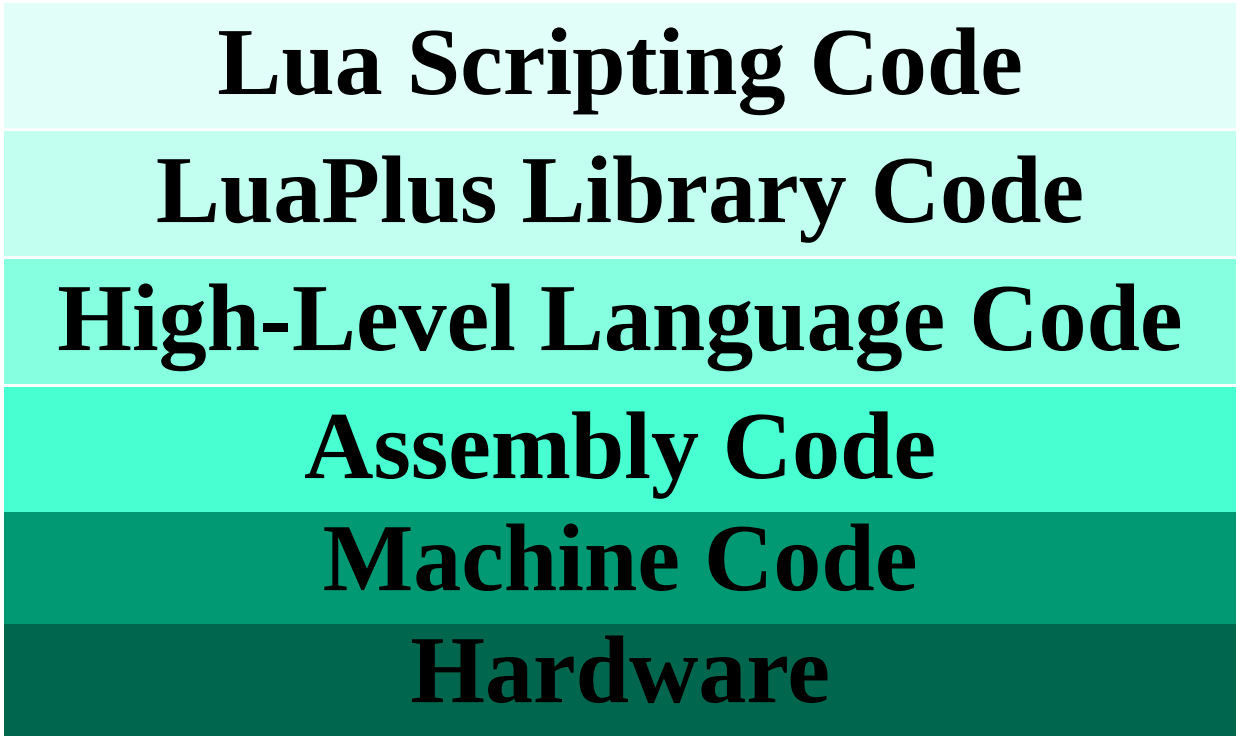
CSE 380 – Computer Game Programming
Game Scripting with LuaPlus

Lua+

What's LuaPlus?

- A C++ framework for using Lua
- Ref: http://luaplus.org/



A practical Example

- Download LuaExample.zip
- Load the solution
- Put a breakpoint on first line
- Start program and watch execution

Let's learn to do some useful things

- C++ → Lua
  - pull a global variable
  - call a function

- Lua → C++
  - call a method

LuaState

- Everything starts here
- Represents a Lua execution environment
- Multiple states allowed
  - each with own global variables
  - good for multithreaded scripting
- First, make a state. Ex:

```
LuaState* luaPState = LuaState::Create();
```

LuaState is our medium

- Has many methods you'll need. Like:
  - **`DoFile`** opens a lua script file
  - **`DoString`** executes a String as Lua code
- Ex:

```
// Open the Lua Script File
int result = luaPState->DoFile("TestScript.lua");

// After this code, there will be a new global variable
// called x, which is a table with 2 elements, which
// could then be used by the lua file's functions
luaPState->DoString("x = { 2, 4 }");
```

Accessing Lua Globals

- We just created a global Lua variable

```
// Let's get it back and see
LuaObject xObj = luaPState->GetGlobal("x");
LuaObject x1Obj = xObj.GetByIndex(1);
int x1 = x1Obj.GetInteger();
cout << x1 << endl;
```

- Did you catch that?
  - we just got data from the Lua script

LuaObject

- Represents a single Lua variable.
  - Can be number, string, table, function, nil, etc.
    - type can be checked via Type method\
- Has conversion methods:
  - **`GetInteger`**
  - **`GetString`**
  - **`Etc.`**
- Assignment methods:
  - **`AssignInteger`**
  - **`AssignString`**
  - **`Etc.`**
- And Test methods:
  - **`IsInteger`**
  - **`IsString`**
  - Etc.

LuaObjects and Tables

- Allows access to data structure
  - remember, can be array or map

- Access provided via:
  - **`GetByName`**
  - **`GetByObject`**
  - **`GetByIndex`**

Let's get a script's variable

- Look at TestScript.lua, you'll see:

```
health = 100
```

- To get this from C++:

```
// Get a global variable
LuaObject healthObj = luaPState->GetGlobal("health");
int health = healthObj.GetInteger();
cout << health;
cout << endl;
```

We can even change the value

- From C++:

```
healthObj.AssignInteger(luaPState, 200);
health = healthObj.GetInteger();
cout << health;
cout << endl;
```

- What good would this do?

- Lua functions would use the new value

Speaking of functions

- This is kind of the whole point
  - C++ code calling Lua functions
  - Lua code calling C++ functions

- Look at TestScript.cpp:

```
function sq(val)
   return val * val
end
```

- To call from C++, use a LuaFunction:

```
// Let's call the TestScript.lua's Sq function
LuaFunction<float> luaSq = luaPState->GetGlobal("sq");
float sq = luaSq(health);
cout << sq << endl;
```

And in reverse?

- Bind your function to a Lua variable
  - that variable provides access to the script
  - Register method in C++. Ex, in C++:
  - name it for the Lua script
  - invoke from script

- Ex, in script:
  - incAndReturn is C++

```
num = 0

function setNum(initNum)
        num = initNum
end

function foo()
        num = incAndReturn(num)
        num = incAndReturn(num)
        num = incAndReturn(num)
end
```

Some helper methods

```
int LuaIncAndReturn(int num)
{
   num++;
   return num;
}

void getAndPrintNum(LuaState *ls)
{
   // Get a global variable
   LuaObject numObj = ls->GetGlobal("num");
   int num = numObj.GetInteger();
   cout << num;
   cout << endl;
}
```

Lua Script calls C++ function Example

```
// Let's make the IncAndReturn method available
// to be invoked by our script
luaPState->GetGlobals().RegisterDirect("incAndReturn",
   LuaIncAndReturn);

// Let's set a global lua variable we'll use
LuaFunction<int> luaSetNum = luaPState->GetGlobal("setNum");
luaSetNum(5);
getAndPrintNum(luaPState);

// Now let's call a function that calls our method
LuaFunction<void> luaFoo = luaPState->GetGlobal("foo");
luaFoo();
getAndPrintNum(luaPState);
luaFoo();
getAndPrintNum(luaPState);
luaFoo();
getAndPrintNum(luaPState);
```

So what would one do with this?

- To start, build a Scripting System to:
  - manage script loading
  - manage script interactions
  - encapsulate all scripting
  - a go-between for other systems (like AI)

- Common strategies:
  - build common premade functionality in C++
  - provide access to Lua scripts
  - use customized combinations from scripts

How should *you* use Lua?

- For some practical purpose
  - it's up to you what that is

- Note, you must use it as both a:
  - Data Definition Language

  AND
  - Runtime Scripting Language

- Recommendation:
  - use it to control bots

Reference

- **Game Engine Architecture by Jason Gregory**
  - Chapter 14: Runtime Gameplay Foundation Systems

- **Game Coding Complete, 4th Edition by Mike McShaffry and David Graham**
  - Chapter 12: Scripting with Lua