# AI for games

Lecture 7

Scripting

# Organiser

- What is a scripting system?
- Introduction to Lua
- Connecting to Lua through C++

# What is a scripting system?

- Separates games content from the engine
  - Allows creative types more freedom
  - Stops them hassling programmers
  - Lets them feel like programmers
- Reduces dependency
  - Shorter compile times
  - Easier to modify (on the fly?)

# What is a scripting system?

- Interpreted
  - Slower
  - Bigger
  - Easier to change
  - Accessible to modders
- Compiled
  - Faster
  - Smaller
  - Needs to be compiled
  - Inaccessible to modders

# Uses - Dialogue

- Allows you to specify dialogue trees easily.

```
Speak (“Halt stranger. Where are you going?”)
Options(1, “None of your damn business!”,
        2, “I bear an urgent letter for the king.”)
choice = GetChoice()
if(choice == 1)
  Speak(“Churlishly spoken. Thou shalt not pass.”)
elseif(choice==2)
  Speak(“Can it be you have not heard? The king is dead and the
  kingdom is now ruled by his trusted advisor, Sir Mordred.”)
```

# Uses – Stage direction

- Good for scripted events in cutscenes or during in-game storyline progression.

```
Place(wizard1, 50, 200);
SetDestination(wizard1, 500, 500);
PlaceCamera(600,600, 50);
Wait(15);
PlayAnimation(wizard1, conversation);
PlaySound(speech_02);
```

# Uses – Mission flow

- Often uses lots of simple scripts at various places in the world.

```
function door_27
  if(mission6_stage == 6)
       Message(“You have arrived at the castle.”)
       Diary(“I reached castle Doom, but Lady Medraint was
  not there to greet me. I began my search of the
  castle.”)
       mission6_stage ==7
  end
end
```

# Uses – AI Logic

- Since this often needs lots of tinkering you can write states in a script.
  - Possibly transfer them to program code once they are working.

```
state attacking
  if(range > 500)
       ChangeState(pursue)
  if(range < 50 and weapons[2]==true)
       SwitchWeapon(2)
  if(health<20)
       ChangeState(flee)
end state
```

# Introduction to Lua

- Used in
  - Baldur's Gate
  - Homeworld 2
  - Far Cry
  - Escape from Monkey Island

# Introduction to Lua

- Lua is
  - Goddess of captured weapons
  - Interpreted
  - Written in C
  - Dynamically typed
  - Case sensitive
- Lua has
  - an interactive interpreter to try out scripts
  - garbage collection

# Introduction to Lua

- Lua was created by
  - the Computer Graphics Technology group
- at the
  - Pontifical Catholic University, Brazil.

# Introduction to Lua

- Variables can be
  - nil
  - number
  - string
  - table
  - function
  - custom
  - boolean (0 or nil is false everything else is true)

# Introduction to Lua

- Conditions use
  - and or not ==
  - if / else / elseif / end
  - repeat / until
  - for / end
    - for i = 50, 250, 5 do

# Introduction to Lua

- Functions use two alternative syntax versions.

```
GoTo =  function(x,y)
  PlayerX = x;
  PlayerY = y;
End
```

- Or

```
function GoTo(x,y)
  PlayerX = x;
  PlayerY = y;
end
```

# Introduction to Lua

- Functions can return more than one thing:

```
function GetStats()
  return health, fatigue
end

h, f = GetStats();
```

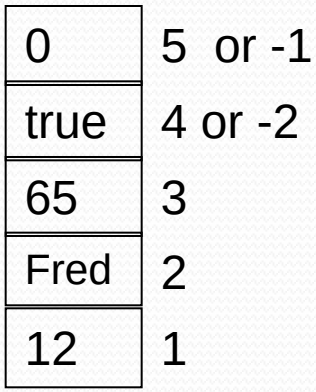
# Introduction to Lua

```
-- This is a comment
```

# Connecting to C++

- To connect with C++ you can:
  - Access Lua global variables
  - Run a Lua function
  - Run a C++ function in Lua
  - Use Luabind
- Connecting with C++ uses a “virtual stack”.

# Connecting to C++



# Connecting to Lua

```
// Include Lua headers and libraries
extern"C"
{
  #include <lua.h>
  #include <lualib.h>
  #include <lauxlib.h>
}

#pragma comment(lib, "libs/lua5.1.lib")
```

# Connecting to Lua

```
// Connect to Lua
  lua_state* pLuaState = lua_open();

// Open Lua's own libraries if needed
  luaopen_base(pLuaState);
  luaopen_string(pLuaState);
  luaopen_math(pLuaState);
  luaopen_io(pLuaState);
  luaopen_table(pLuaState);
```

# Connecting to Lua

```
// Load a file
  luaL_dofile(pLuaState, "botscript.lua");

// Don’t forget to tidy up at end of game
  if(pLuaState)
      lua_close(pLuaState);
```

# Accessing global variables

```
// Empty the stack if needed
lua_settop(pLuaState,0);

// Get the variable
lua_getglobal(pLuaState, "currentMission");

// Check valid
if(!lua_isnumber(pLuaState,1))
  {
      // error
  }
```

# Accessing global variables

```
// Convert to C++ variable
int cm = (int)lua_tonumber(pLuaState,1);

// Clear stack
lua_pop(pLuaState);
```

# Accessing Lua functions

```
// Get the function onto the stack
lua_getglobal(pLuaState, "GetDest");

// Check it is valid
if(!lua_isfunction(pLuaState, -1))
{
  // error
}
```

# Accessing Lua functions

```
// Put parameters onto the stack
  int charNumber = 2;
  double time = 12.0;
  lua_pushnumber(pLuaState, charNumber);
  lua_pushnumber(pLuaState, time);

// Call the function (giving number of
// parameters and returns)
  lua_call(pLuaState, 2,2);
```

# Accessing Lua functions

```
// Extract the returns from the stack
destX = lua_tonumber(pLuaState, -2);
destY = lua_tonumber(pLuaState, -1);
```

# Accessing C++ functions from Lua

- Make a valid C++ function that Lua can call.
  - Must be int return and take lua_state* as a parameter.

```
int GetFireSolution (lua_state* pLS)
{
```

# Accessing C++ functions from Lua

- The function will get parameters from the stack

```
int GetFireSolution(lua_state* pLS)
{
  // should check valid, as usual
  double x = lua_tonumber(pLS, 1);
  double y = lua_tonumber(pLS, 2);
```

# Accessing C++ functions from Lua

- Function will stick the return(s) onto the stack

```
    lua_pushnumber(pLS, angle);
    lua_pushnumber(pLS, velocity);
    lua_pushnumber(pLS, bearing);

    // return the number of returns
    return 3;
}
```

# Accessing C++ functions from Lua

- Before Lua calls the function, the function must be registered with Lua

```
lua_register(pLuaState, “GetFireSolution”,
GetFireSolution);
```

# Summary

- Scripting
  - 4 uses
- Potted Lua
  - Types, functions, structures
- Interfacing
  - Globals
  - Lua functions
  - C++ functions