



# 3D Game Programming 51

## -Game Programming with Lua

[afewhee@gmail.com](mailto:afewhee@gmail.com)





- Lua - 개요
- Lua Embedding
- Game Interface
- Lua 문법
- Lua 게임 제작



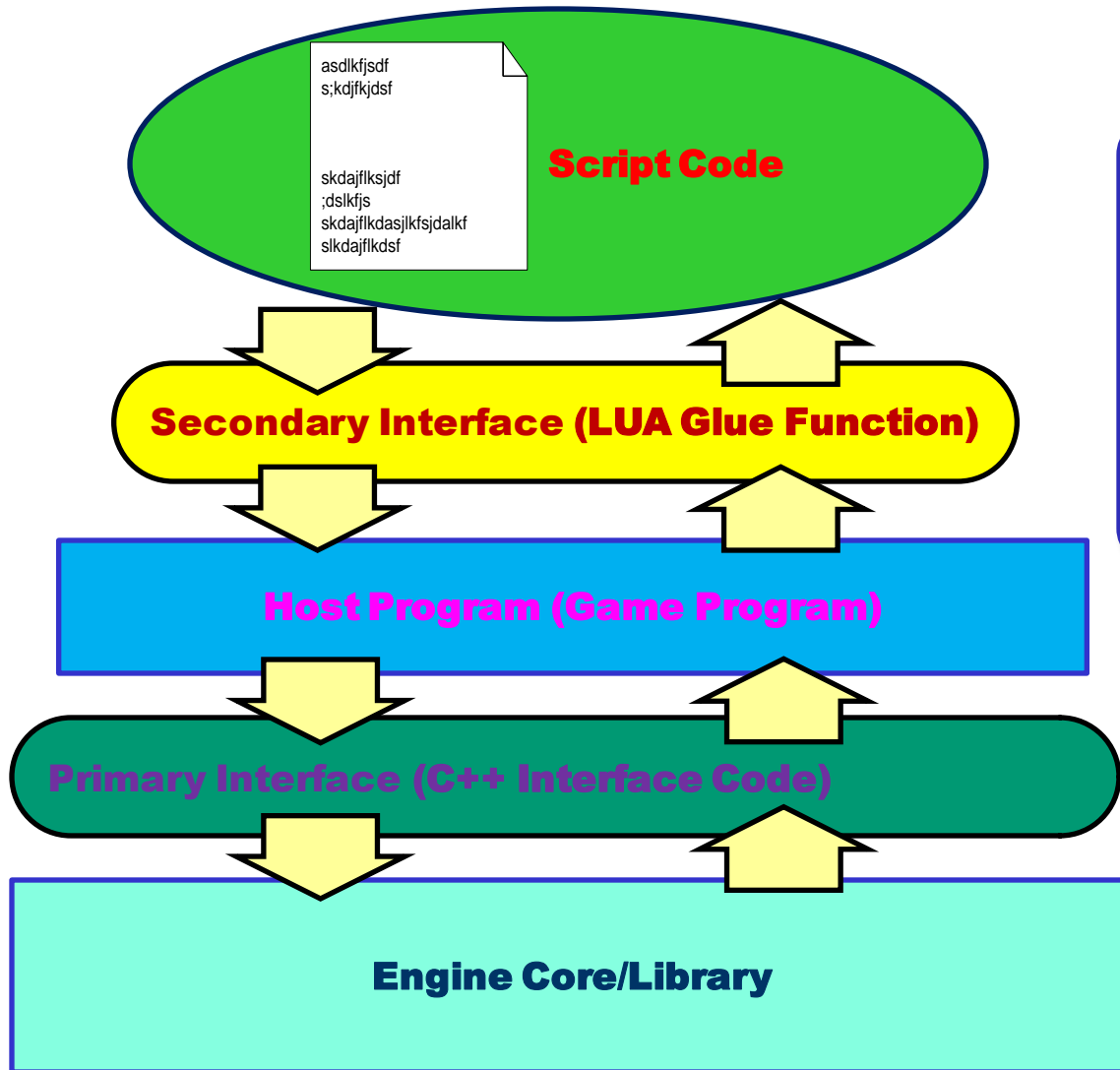


- 스크립트 언어란?
  - ◆ 인터프리터에 의해 번역
  - ◆ 컴파일 되지 않은 언어 → 번역 즉시 실행
  - ◆ 인터프리터 필요
- 스크립트 언어의 목적
  - ◆ 개발 기간 및 개발 비용 단축
  - ◆ 비 숙련성
  - ◆ 자주 갱신해야 하는 복잡한 프로그램
- 스크립트 언어들
  - ◆ 웹: 펄, PHP, ASP, JSP, VBA, 자바스크립트
  - ◆ 게임: 루아, 파이썬





## ● 게임 시스템에서 Lua의 위치



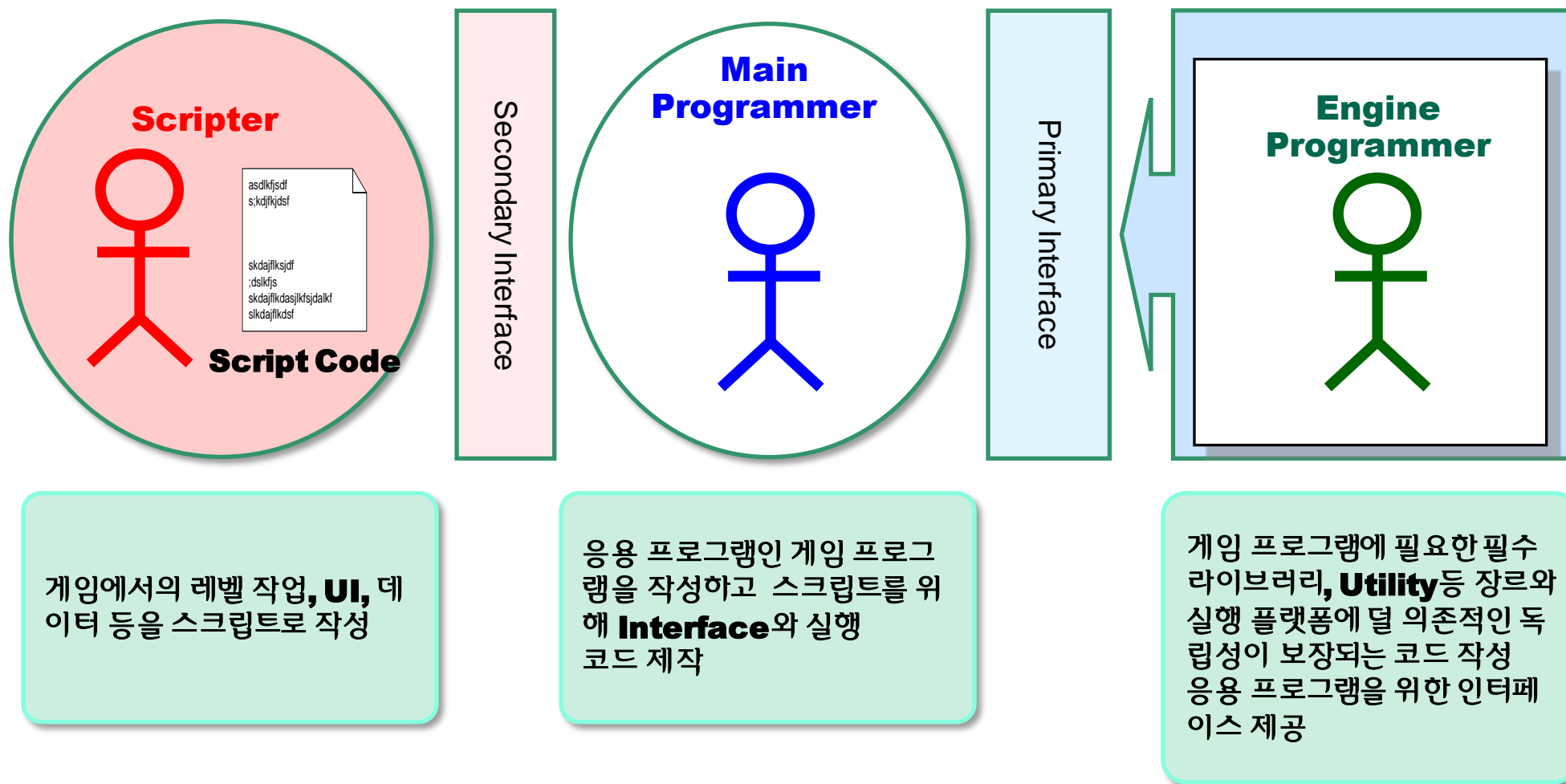
**Script** 코드는 응용 프로그램과 인터페이스를 통해서 통신

응용 프로그램은 스크립트 해석에 대한 인터프리터, 실행에 대한 코드 포함



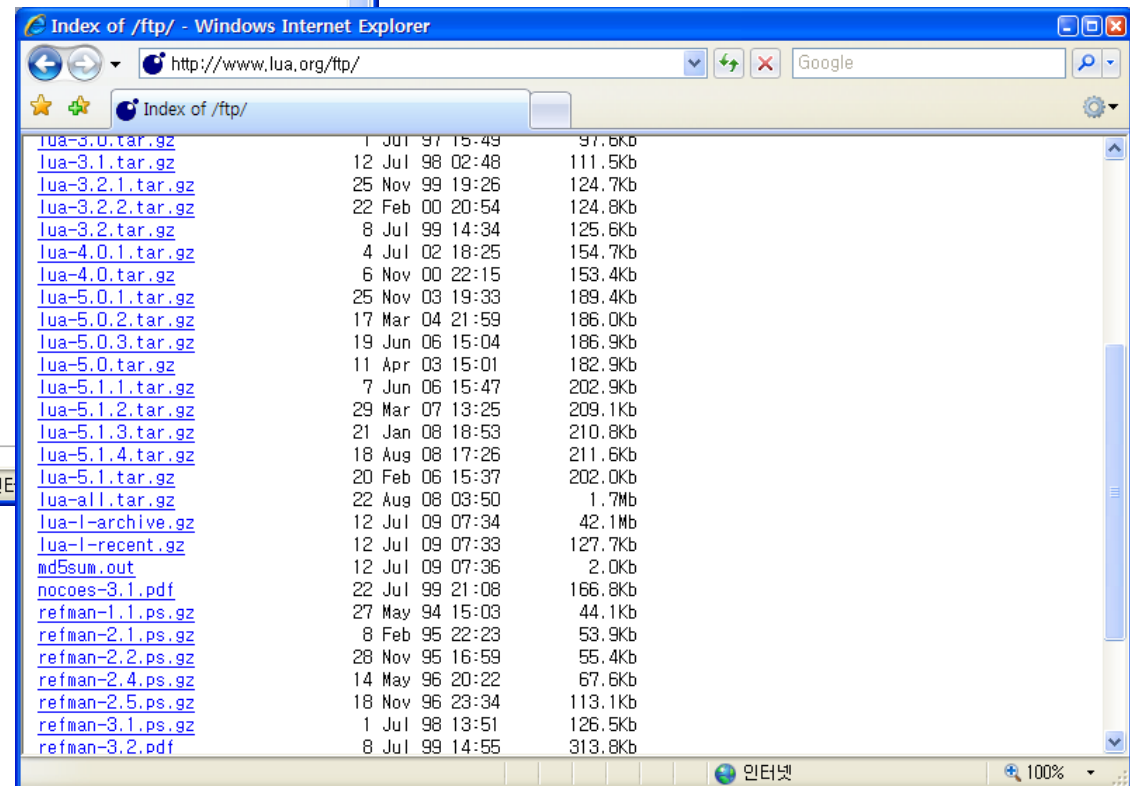
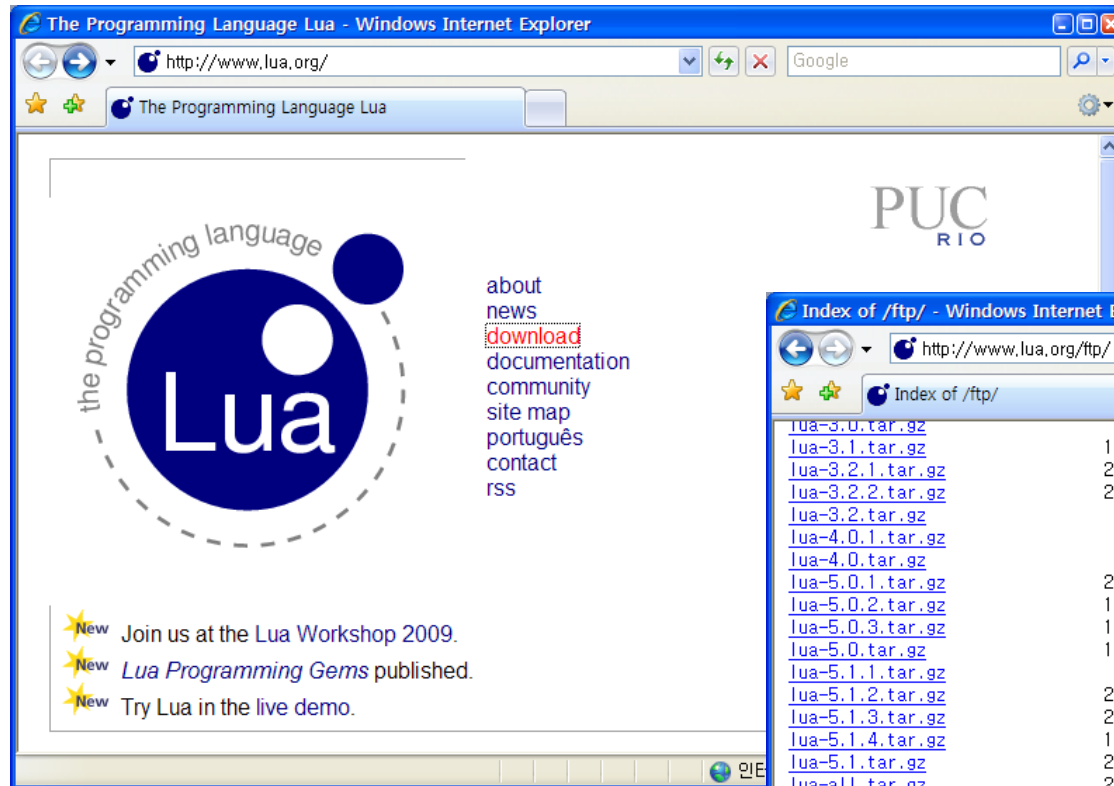


## ● 게임 개발자 관계



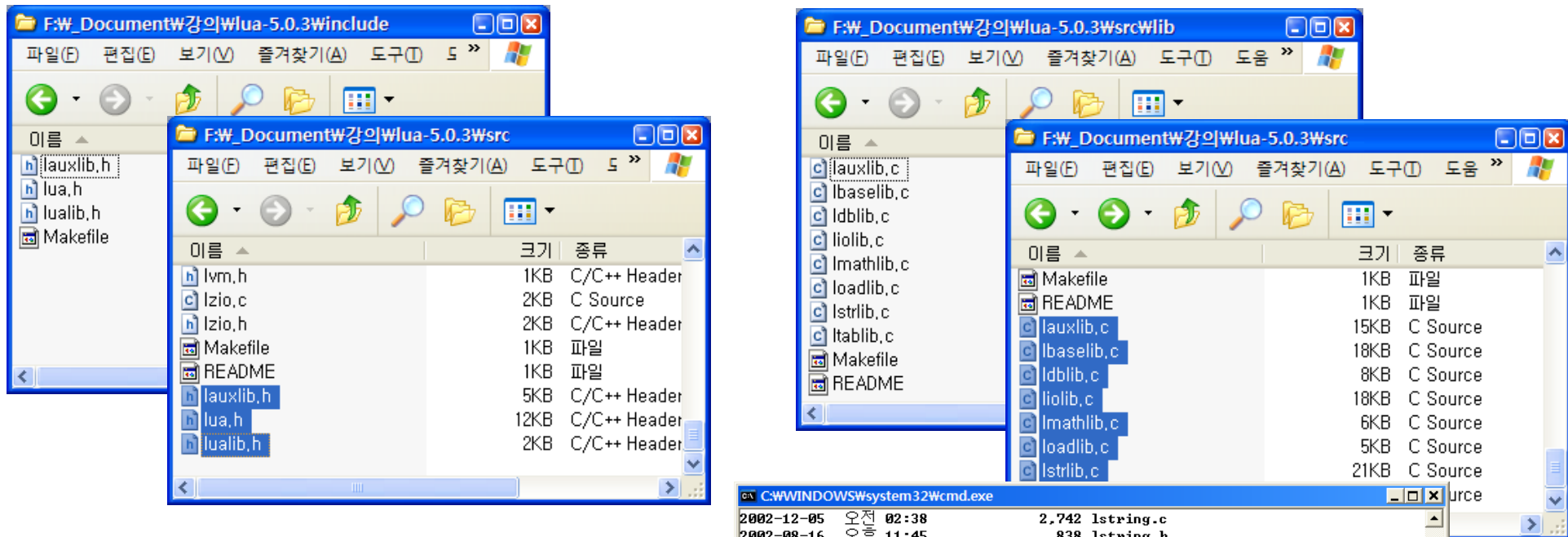


### ● Source Down Load

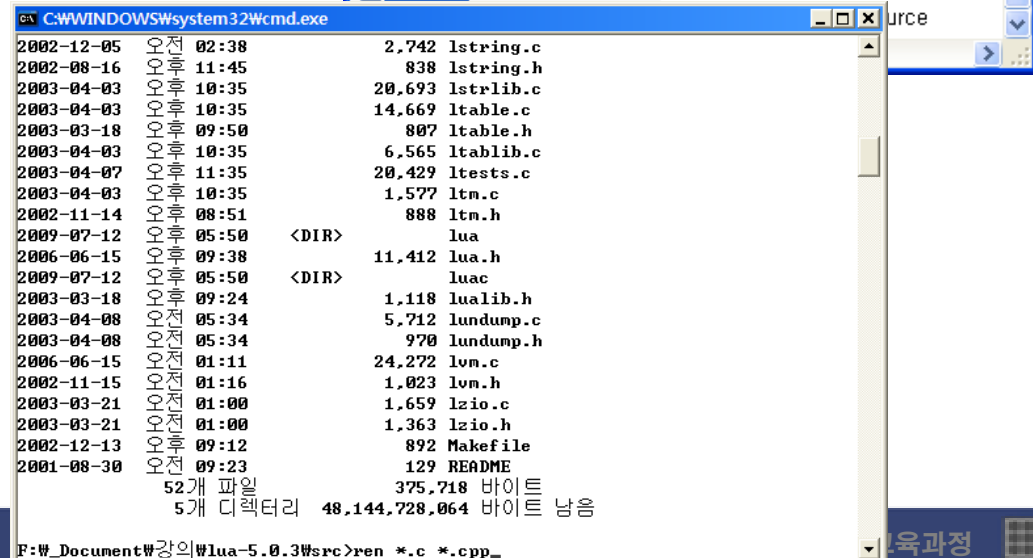


## 2. Lua Embedding - Compile

- include/ .h, src/lib/\*.cpp, .h 파일 옮기기

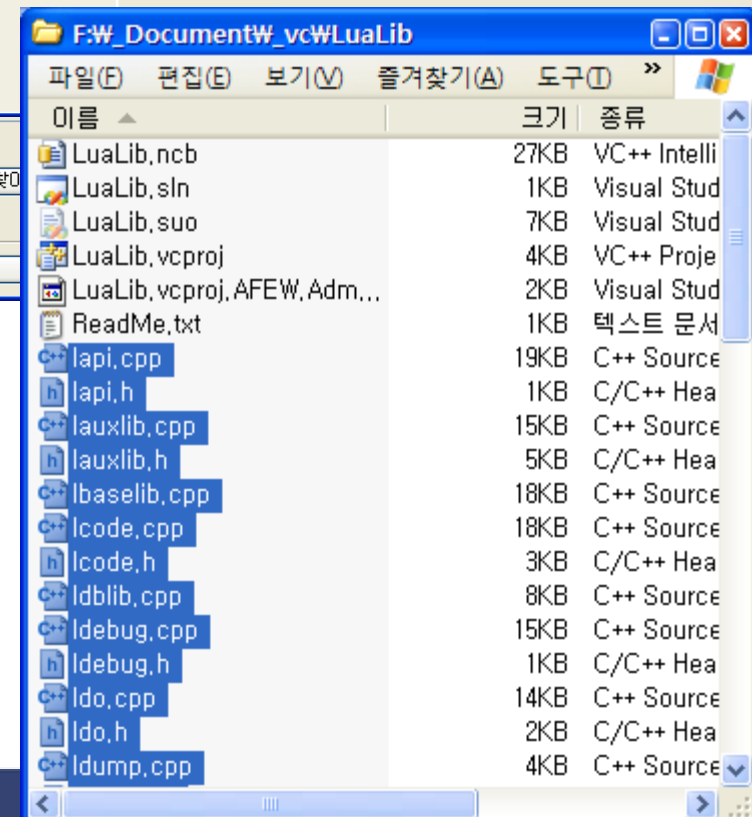
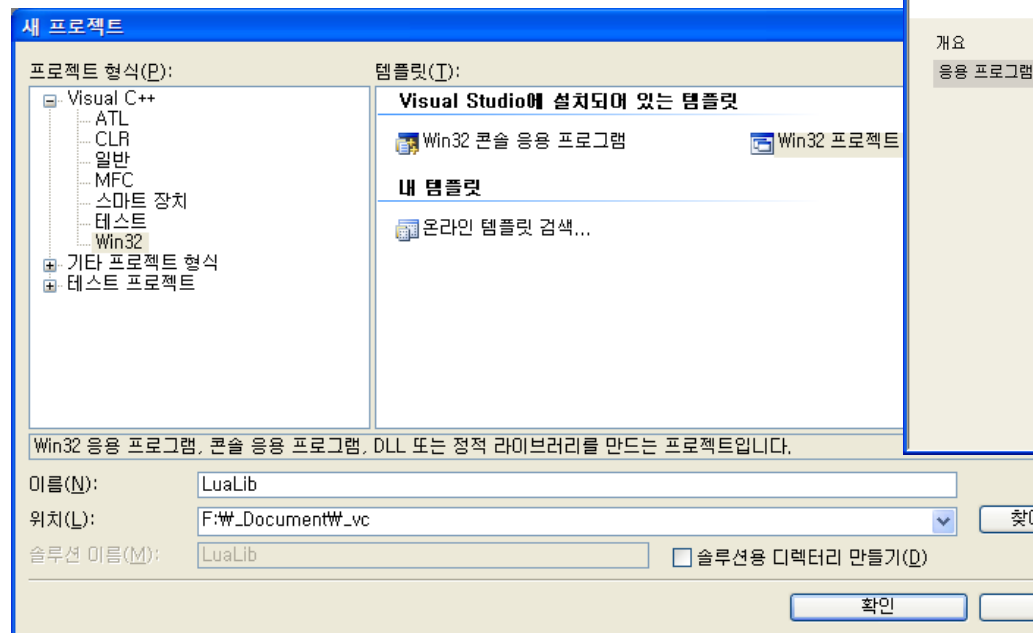


- 파일 확장자 이름 변경
  - ◆ cpp로 컴파일





### ● Lua Library 만들기



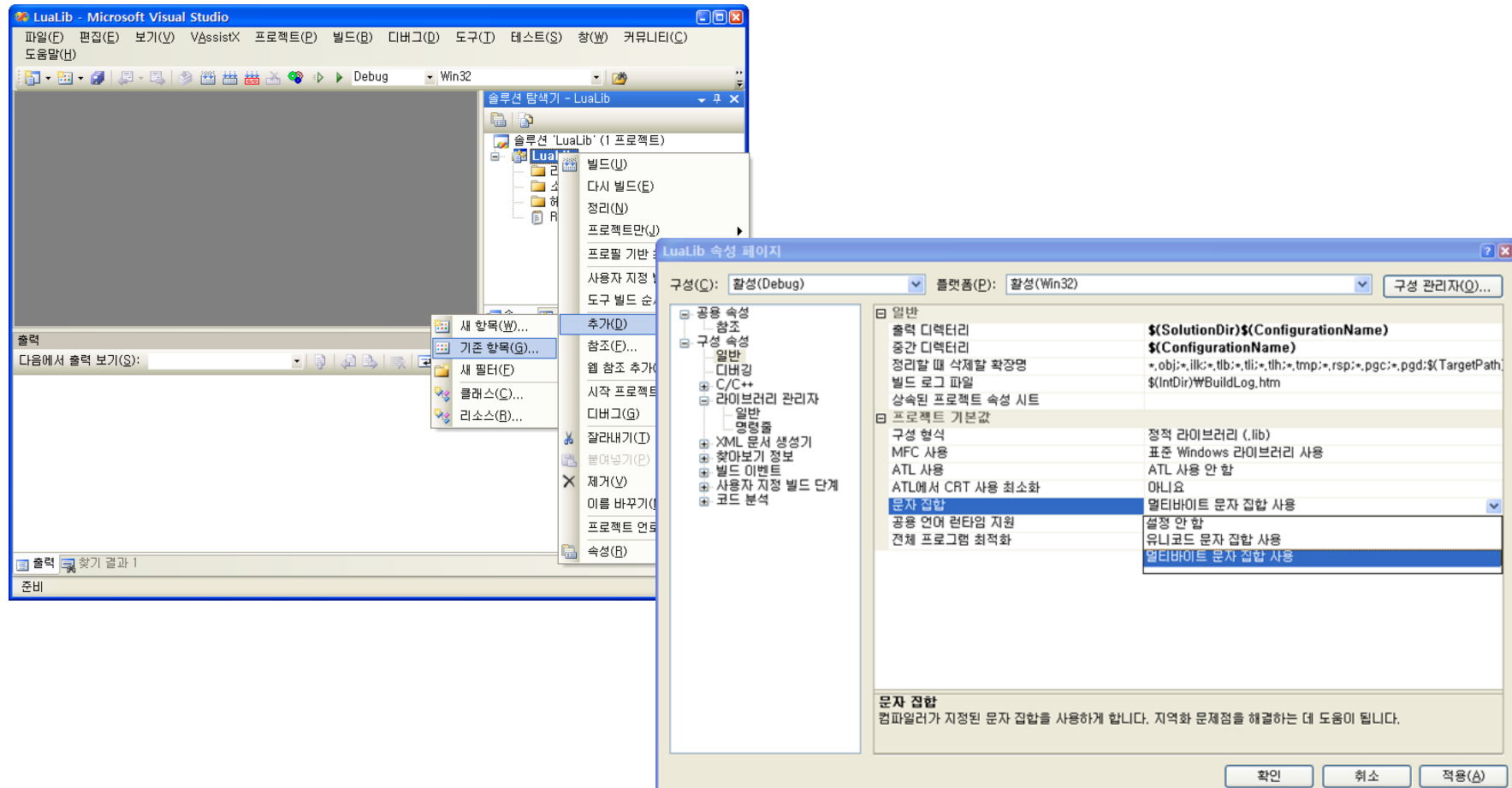
### ● Lua 파일을 라이브러리 폴더에 옮기기 ltests.c 파일 제외





### ● Lua Library Compile

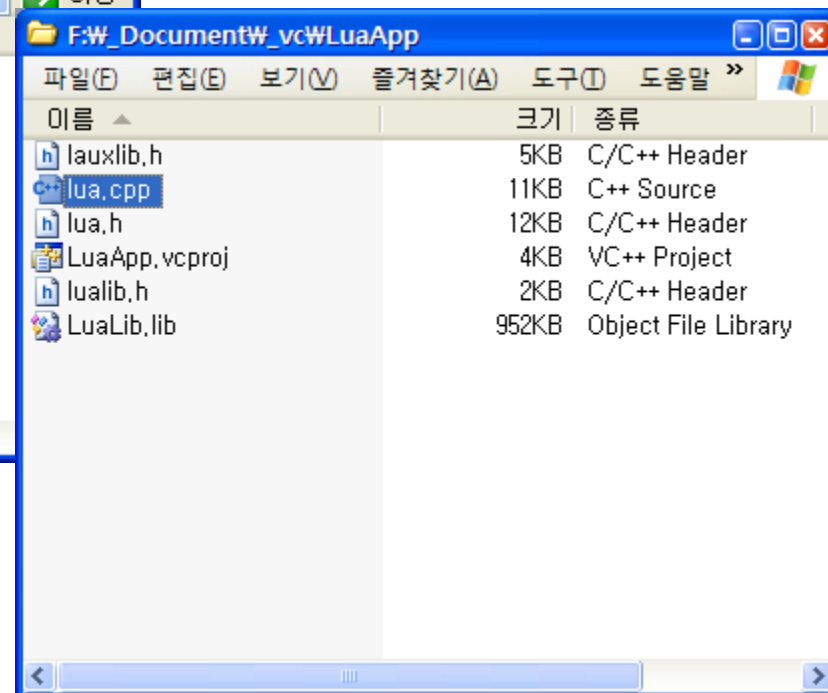
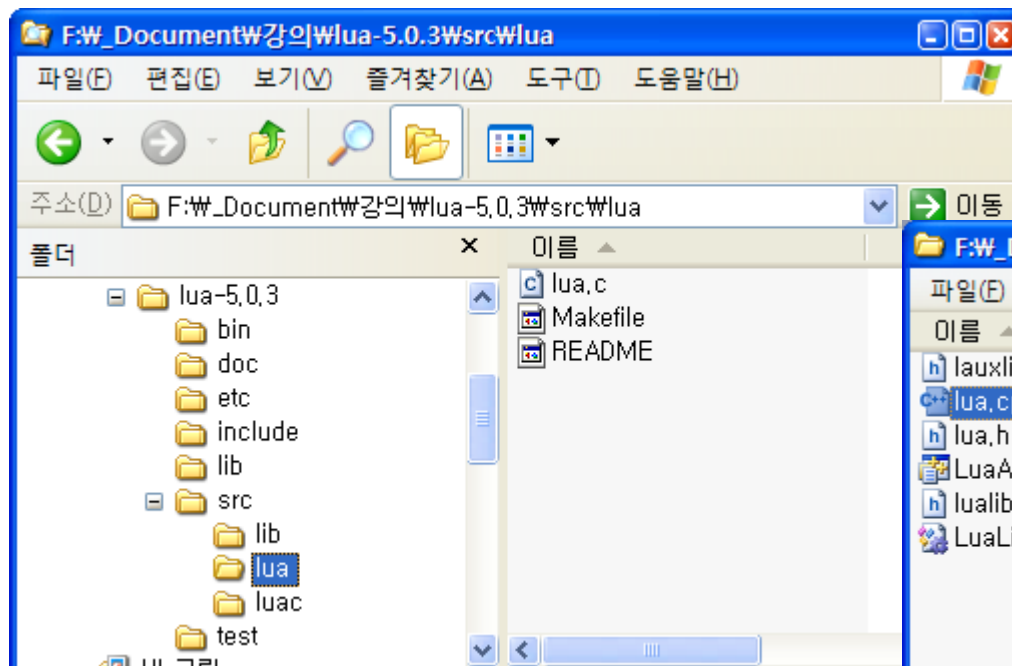
#### ◆ Multi-Byte Code 사용





### ● Library Test

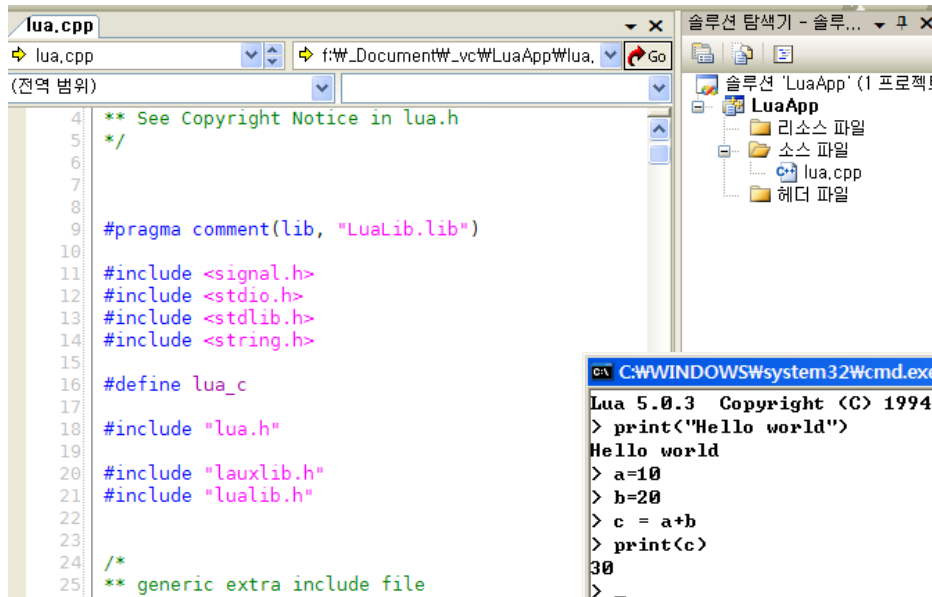
◆ lua.c → lua.cpp로 변경한 다음 콘솔 프로젝트 작성



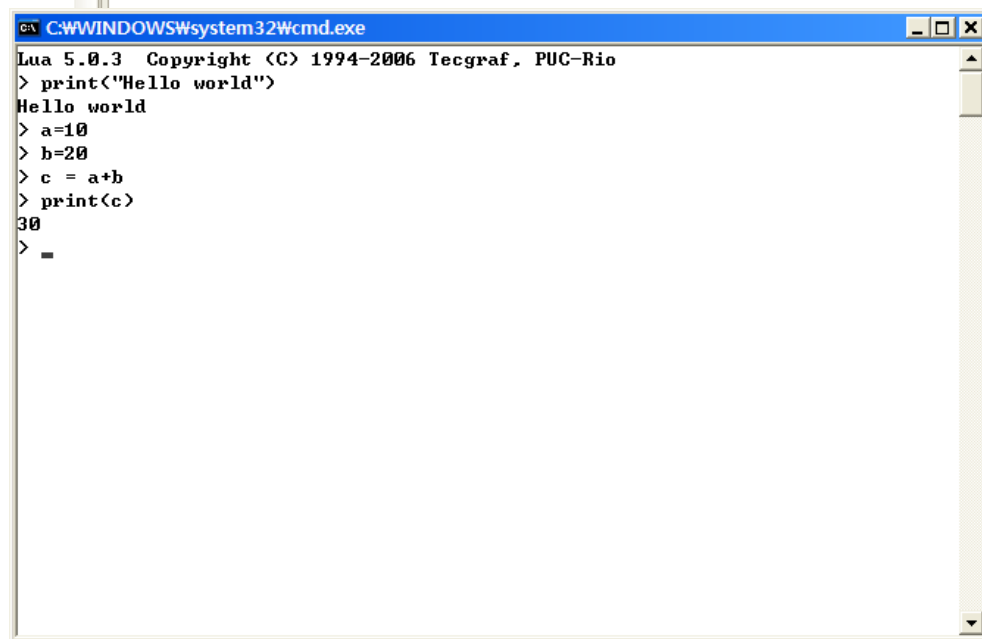


### ● Library Test

◆ lua.cpp에 라이브러리 링크 지시한 다음 컴파일



```
lua.cpp
lua.cpp
f:\W_Document\vc\WLuaApp\lua,
(전역 범위)
4  ** See Copyright Notice in lua.h
5  */
6
7
8
9  #pragma comment(lib, "LuaLib.lib")
10
11  #include <signal.h>
12  #include <stdio.h>
13  #include <stdlib.h>
14  #include <string.h>
15
16  #define lua_c
17
18  #include "lua.h"
19
20  #include "lauxlib.h"
21  #include "luaLlib.h"
22
23
24  /*
25  ** generic extra include file
```



```
C:\WINDOWS\system32\cmd.exe
Lua 5.0.3 Copyright (C) 1994-2006 Tecgraf, PUC-Rio
> print("Hello world")
Hello world
> a=10
> b=20
> c = a+b
> print(c)
30
>
```



## 2. Lua Embedding - Compile

### ● Modular 연산자 "%" 추가

- ◆ %연산자와 가장 근접한 나누기 연산자 키워드 div를 찾는다.
- ◆ 제일 먼저 찾은 lopcodes.cpp파일의 luaP\_opnames 변수 안에 있는 "DIV"(35 Line) 값 아래 "MOD"를 추가한다.
- ◆ 또한 lopcodes.cpp의 81 라인 OP\_DIV 정의 아래 OP\_DIV와 똑 같이 복사해서 붙인다.
  - ,opcode(0, 0, 1, 1, 1, 0, iABC) /\* OP\_MOD \*/
- ◆ 이 때 const lu\_byte luaP\_opmodes[NUM\_OPCODES] 변수의 크기가 NUM\_OPCODES으로 되어 있으므로 NUM\_OPCODES가 정의 되어 있는 lopcodes.h로 이동한다.
- ◆ 화면을 위로 올리면 155라인에 OP\_DIV가 있다. 마찬가지로 이 부분을 복사해서 OP\_MOD를 붙인다.

찾기 및 바꾸기

파일에서 찾기 | 빠른 바꾸기

찾을 내용(N):  
div

찾는 위치(L):  
전체 솔루션

☒ 하위 폴더 포함(B)

☒ 찾기 옵션(O)

☐ 대/소문자 구분(C)

☐ 단어 단위로(W)

☐ 사용(E):

정규식

이 파일 형식 보기(I):

☒ 결과 옵션(S)

모두 찾기

```

19: const char *const luaP_opnames[] =
20:     "MOVE",
21:     "LOADK",
22:     "LOADBOOL",
23:     "LOADNIL",
24:     "GETUPVAL",
25:     "GETGLOBAL",
26:     "GETTABLE",
27:     "SETGLOBAL",
28:     "SETUPVAL",
29:     "SETTABLE",
30:     "NEWTABLE",
31:     "SELF",
32:     "ADD",
33:     "SUB",
34:     "MUL",
35:     "DIV",
36:     "MOD",
37:     "POW",
38:     "UNM",

```

```

64: const lu_byte luaP_opmodes[NUM_OPCODES] =
65: /*
66:  T  B  Bk Ck  sA  K  mode
67:  ,opcode(0, 1, 0, 0, 1, 0, iABC)
68:  ,opcode(0, 0, 0, 0, 1, 1, iABx)
69:  ,opcode(0, 0, 0, 0, 1, 0, iABC)
70:  ,opcode(0, 1, 0, 0, 1, 0, iABC)
71:  ,opcode(0, 0, 0, 0, 1, 0, iABC)
72:  ,opcode(0, 0, 0, 0, 1, 1, iABx)
73:  ,opcode(0, 1, 0, 1, 1, 0, iABC)
74:  ,opcode(0, 0, 0, 0, 0, 1, iABx)
75:  ,opcode(0, 0, 0, 0, 0, 0, iABC)
76:  ,opcode(0, 0, 1, 1, 0, 0, iABC)
77:  ,opcode(0, 0, 0, 0, 1, 0, iABC)
78:  ,opcode(0, 1, 0, 1, 1, 0, iABC)
79:  ,opcode(0, 0, 1, 1, 1, 0, iABC)
80:  ,opcode(0, 0, 1, 1, 1, 0, iABC)
81:  ,opcode(0, 0, 1, 1, 1, 0, iABC)
82:  ,opcode(0, 0, 1, 1, 1, 0, iABC)
83:  ,opcode(0, 0, 1, 1, 1, 0, iABC)

```

```

131: typedef enum {
132: /*
133:  name      args      description
134:  -----
135:  OP_MOVE, /* A B R(A) := R(B)
136:  OP_LOADK, /* A Bx      R(A) := Kst(Bx)
137:  OP_LOADBOOL, /* A B C  R(A) := (Bool)B; if (C) PC++
138:  OP_LOADNIL, /* A B R(A) := ... := R(B) := nil
139:  OP_GETUPVAL, /* A B R(A) := UpValue[B]
140:
141:  OP_GETGLOBAL, /* A Bx      R(A) := Gbl[Kst(Bx)]
142:  OP_GETTABLE, /* A B C      R(A) := R(B)[RK(C)]
143:
144:  OP_SETGLOBAL, /* A Bx      Gbl[Kst(Bx)] := R(A)
145:  OP_SETUPVAL, /* A B UpValue[B] := R(A)
146:  OP_SETTABLE, /* A B C      R(A)[RK(B)] := RK(C)
147:
148:  OP_NEWTABLE, /* A B C      R(A) := {} (size = B,C)
149:
150:  OP_SELF, /* A B C      R(A+1) := R(B); R(A) := R(B)[RK(C)]
151:
152:  OP_ADD, /* A B C      R(A) := RK(B) + RK(C)
153:  OP_SUB, /* A B C      R(A) := RK(B) - RK(C)
154:  OP_MUL, /* A B C      R(A) := RK(B) * RK(C)
155:  OP_DIV, /* A B C      R(A) := RK(B) / RK(C)
156:  OP_MOD, /* A B C      R(A) := RK(B) % RK(C)
157:

```



### ● Modular 연산자 "%" 추가

- ◆ lparser.cpp 파일로 옮겨가서 765 라인에서 '/' 연산자 정의 밑에 나누기와 동일하게 '%' 와 OPR\_MOD를 추가한다.
- ◆ OPR\_MOD은 정의 되어 있지 않으므로 OPR\_DIV 정의가 있는 lcode.h 파일 27 라인으로 가서 OPR\_MOD를 정의한다.
- ◆ ltm.cpp 32 라인으로 가서 "\_\_div" 뒤에 "\_\_mod"를 추가한다.
- ◆ lvm.cpp 358 라인의 case TM\_DIV 밑에 TM\_MOD에 대한 코드를 추가한다.
- ◆ lvm.cpp 570 라인 근처 case OP\_DIV 아래에 OP\_MOD를 추가한다.

```
760 static BinOpr getbinopr (int op)
761 {
762     switch (op) {
763         case '+': return OPR_ADD;
764         case '-': return OPR_SUB;
765         case '*': return OPR_MULT;
766         case '/': return OPR_DIV;
767         case '%': return OPR_MOD;
768
769         case '^': return OPR_POW;
```

```
26 typedef enum BinOpr {
27     OPR_ADD, OPR_SUB, OPR_MULT, OPR_DIV, OPR_MOD, OPR_POW,
28     OPR_CONCAT,
29     OPR_NE, OPR_EQ,
30     OPR_LT, OPR_LE, OPR_GT, OPR_GE,
31     OPR_AND, OPR_OR,
32     OPR_NOBINOPR
33 } BinOpr;
```

```
27 L
28 void luaT_init (lua_State *L) {
29     static const char *const luaT_eventname[] = {
30         "__index", "__newindex",
31         "__gc", "__mode", "__eq",
32         "__add", "__sub", "__mul", "__div", "__mod",
33         "__pow", "__unm", "__lt", "__le",
34         "__concat", "__call"
35     };
```

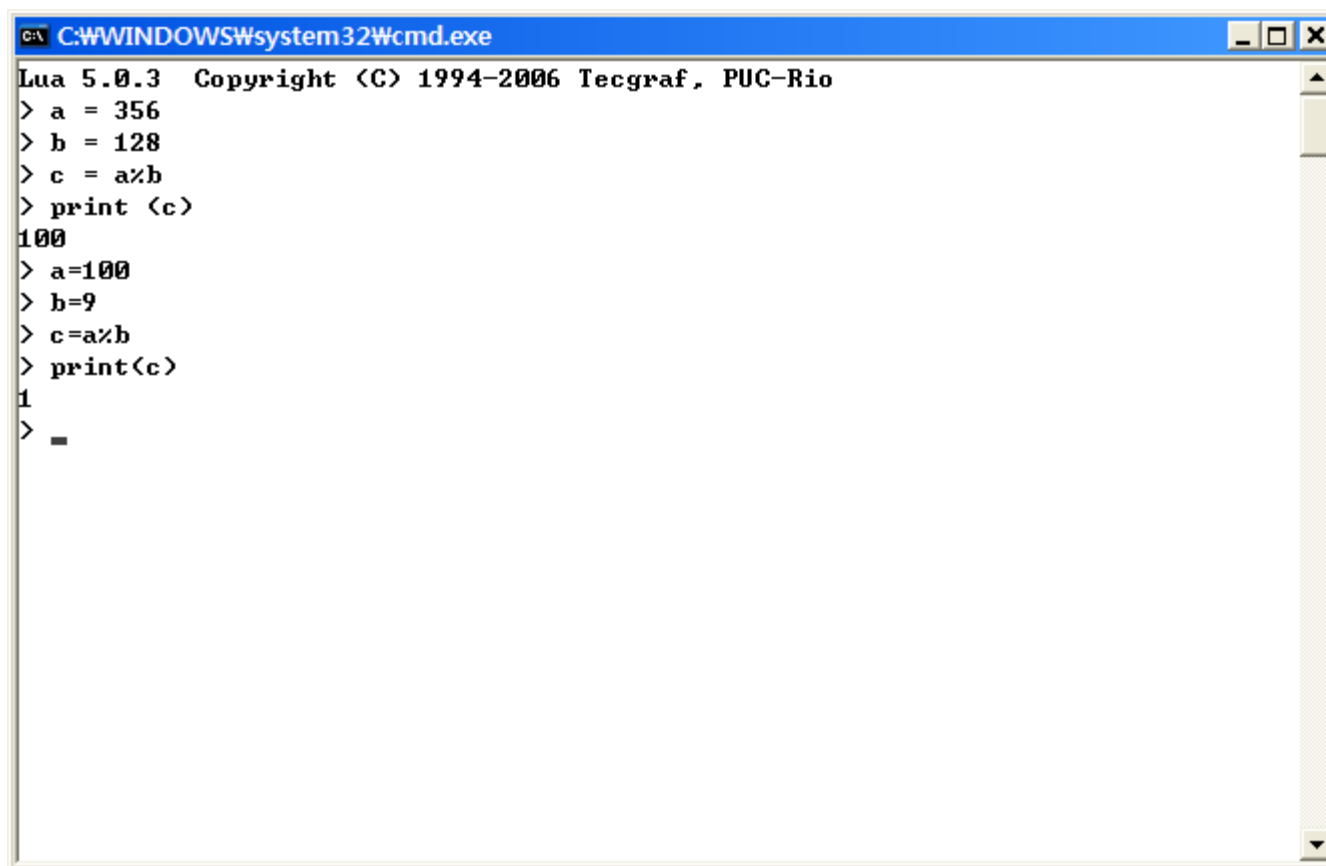
```
360 case TM_MOD:
361 {
362     double db = nvalue(b);
363     double dc = nvalue(c);
364
365     int tb = (int)db;
366     int tc = (int)dc;
367     setnvalue(ra, tb % tc);
368     break;
369 }
```

```
574 case OP_MOD: {
575     TObject *rb = RKB(i);
576     TObject *rc = RKC(i);
577
578     if (ttisnumber(rb) && ttisnumber(rc))
579     {
580         double db = nvalue(rb);
581         double dc = nvalue(rc);
582         int tb = (int)db;
583         int tc = (int)dc;
584         setnvalue(ra, tb % tc);
585     }
586     else
587         Arith(L, ra, rb, rc, TM_MOD);
588     break;
589 }
```





### ● Modular 연산자 "%" Test



```
C:\WINDOWS\system32\cmd.exe
Lua 5.0.3 Copyright (C) 1994-2006 Tecgraf, PUC-Rio
> a = 356
> b = 128
> c = a%b
> print(c)
100
> a=100
> b=9
> c=a%b
> print(c)
1
>
```

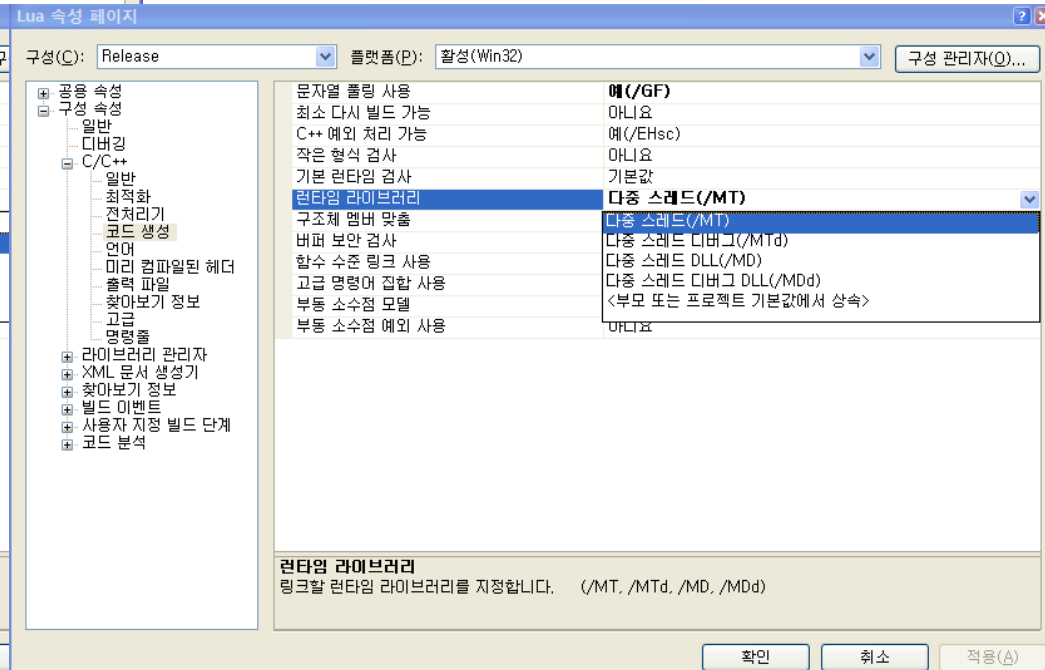
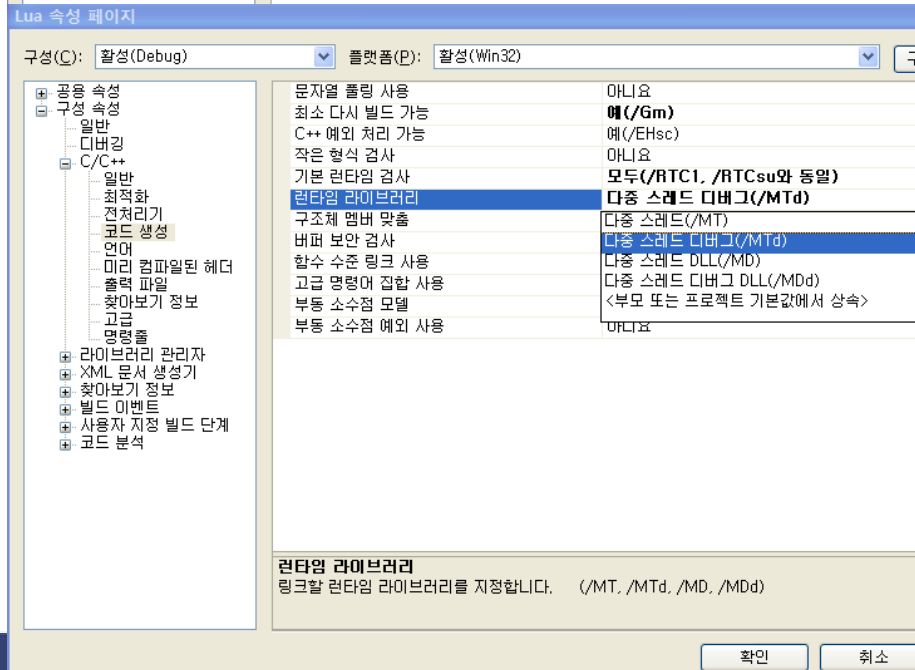
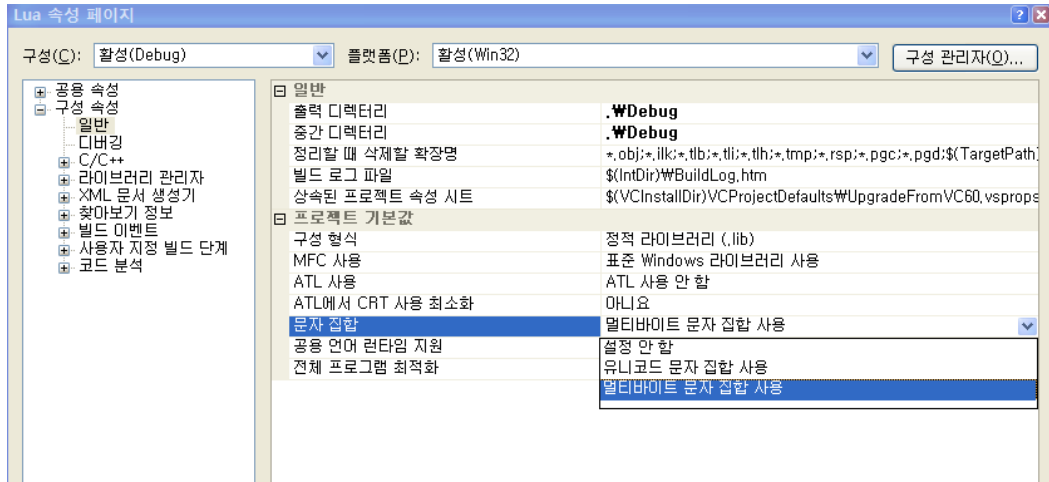




## 2. Lua Embedding – Library Compile Option

### ● Library Compile Option

#### ◆ 반드시 Host Program과 동일하게 설정





- 시작

- ◆ `g_pLuaSt = lua_open();`

- 종료

- ◆ `lua_close(g_pLuaSt);`

- load Lua libraries

- ◆ `luaopen_base(g_pLuaSt);`    // base

- ◆ `luaopen_io(g_pLuaSt);`    // 입출력

- ◆ `luaopen_math(g_pLuaSt);`    // 수학함수

- ◆ `luaopen_string(g_pLuaSt);`    // 문자열

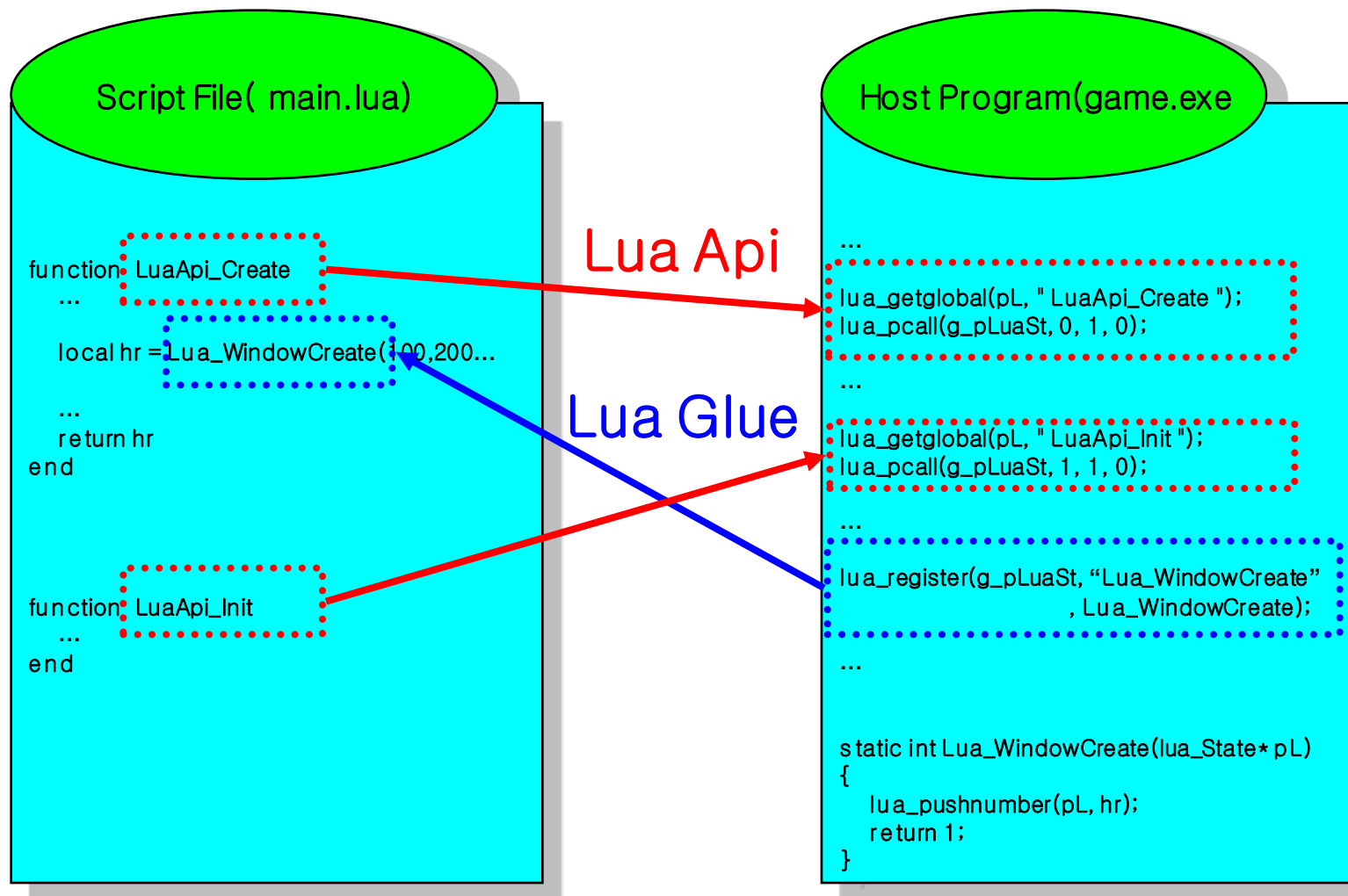
- ◆ `luaopen_table(g_pLuaSt);`    // 루아 테이블







### ● Lua Script와 Host Program (C-base)

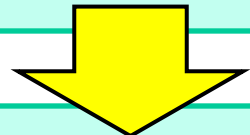




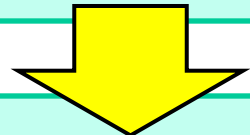
## 2. Lua Embedding - Lua 스크립트 파일의 함수 호출

- 스크립트 로드
  - ◆ lua\_dostring(lua\_State\*, ScriptString);
  - ◆ lua\_dofile(lua\_State\*, "File Name");
  - ◆ lua\_dobuffer(lua\_State\*, buffer, ...);
- Lua 실행 함수 (Lua API) 로드
  - ◆ lua\_getglobal(lua\_State\*, "함수이름");
- Lua API 함수에 인수 전달
  - ◆ lua\_pushnumber(lua\_State\*, v);
  - ◆ lua\_pushstring(lua\_State\*, v);
- 스크립트 함수 호출
  - ◆ lua\_pcall(lua\_State\*, 함수에 전달할 인수 개수, 리턴 개수, 0);
- 리턴 인덱스 받기
  - ◆ int nIdx = lua\_gettop(lua\_State\*);
- 반환 받은 데이터 해석
  - ◆ 반환 인덱스 얻기: nIndex = lua\_gettop();
  - ◆ 숫자, 문자 확인: lua\_isnumber(), lua\_isstring();
  - ◆ 숫자로 변환: lua\_tonumber(lua\_State\*, nIndex);
  - ◆ 문자열로 변환: (char\*)lua\_tostring(lua\_State\*, nIndex);

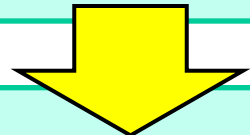
// 스크립트 로드  
**lua\_dofile(..., "파일 이름");**



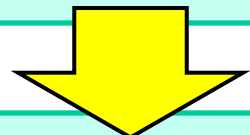
//Lua API 함수 로드  
**lua\_getglobal(..., "루아 함수 이름");**



//인수 전달  
**lua\_pushnumber(..., ...);**  
**lua\_pushstring(...);**



// Lua 함수 실행  
**lua\_pcall(..., 2, 2, 0);**



//Lua 함수 실행 결과 처리  
//반환 인덱스 얻기  
**nIdx = lua\_gettop();**  
**lua\_tonumber(..., nIdx-0);**  
**lua\_tostring(..., nIdx-1);**





## 2. Lua Embedding - Lua API 호출 예

// c-code

```
lua_dofile(m_pLuaSt, "main.lua");
```

```
int a = 100;
```

```
int b = 200;
```

```
lua_getglobal(m_pLuaSt, "Lua_MyGlue");
```

```
lua_pushnumber(m_pLuaSt, a);
```

```
lua_pushnumber(m_pLuaSt, b);
```

```
lua_pcall(m_pLuaSt, 2, 3, 0);
```

```
int nldx = lua_gettop(m_pLuaSt);
```

```
int v1 = (int)lua_tonumber(m_pLuaSt, nldx-0);
```

```
char* v2 = (char*)lua_tostring(m_pLuaSt, nldx-1);
```

```
int v3 = (int)lua_tonumber(m_pLuaSt, nldx-2);
```

```
printf("%d, %s %d\n", v1, v2, v3);
```

--main.lua

```
function Lua_MyGlue(a, b)  
    return a*10, "Hello world", b*10  
end
```

```
C:\WINDOWS\system32\cmd.exe  
2000. Hello world 1000  
계속하려면 아무 키나 누르십시오 . . .
```





## 2. Lua Embedding – 스크립트에서 Lua Glue 사용

- Glue 함수 작성

- ◆ `static int FunctionName(lua_State* pL)`
- ◆ 반환의 int는 스크립트에서 호출될 때 반환 숫자

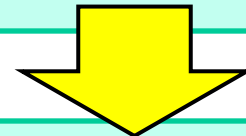
- Glue 함수 등록

- ◆ `lua_register(lua_State*, "루아에서의 이름", 함수);`

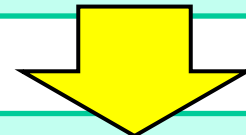
- 스크립트 로드

- ◆ `lua_dofile(lua_State*, "script/sample.lua");`

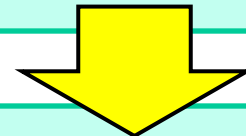
```
static int Mcl_Func(lua_State* pL)  
{  
    lua_pushnumber(...);  
    lua_pushstring(...);  
  
    return 2;  
}
```



```
//Glue 함수 등록  
lua_register(..., "MyFunc", Mcl_Func);
```



```
_ 루아 스크립트  
function Lua_Init()  
    ...  
    MyFunc()  
    ...  
end
```



```
//스크립트 로드  
lua_dofile(...);
```





### //c-code

```
static int Mcl_FontLoad(lua_State* pL) {  
    ...  
    lua_pushnumber(pL, v);  
    return 1;  
}  
  
static int Mcl_FontDestroy(lua_State* pL) {  
    ...  
    return 0;  
}  
  
static int Mcl_FontDraw(lua_State* pL) {  
    return 0;  
}  
...  
// lua_glow 함수 등록  
lua_register(pLuaSt, "Mcl_FontLoad", Mcl_FontLoad);  
lua_register(pLuaSt, "Mcl_FontDestroy", Mcl_FontDestroy);  
lua_register(pLuaSt, "Mcl_FontDraw", Mcl_FontDraw);  
  
lua_dofile(pLuaSt, "main.lua");
```

### --lua file

```
function Lua_Init()  
    font = Mcl_FontLoad(...)  
  
end  
  
function Lua_Destroy()  
    Mcl_FontDestroy(font)  
end  
  
function Lua_Render()  
    Mcl_FontDraw(font, ...)  
end
```





- Application 에서 꼭 호출해야 할 Lua API
  - ◆ 시스템 생성과 초기화 `Lua_Create()`
  - ◆ 데이터 초기화 `Lua_Init()`
  - ◆ 데이터 해제 `Lua_Destroy()`
  - ◆ 데이터 갱신 `Lua_FrameMove()`
  - ◆ 렌더링 `Lua_Render()`
- 스크립트에서 꼭 호출해야 할 Lua Glue(C함수)
  - ◆ 시스템에 따라 정의
  - ◆ 시스템 관련 `Mcl_CreateWindow()`
  - ◆ 게임 기본 객체 `Mcl_MouseEvtnt()`
  - ◆ 기타...





- 텍스처 관리 자료 구조 필요

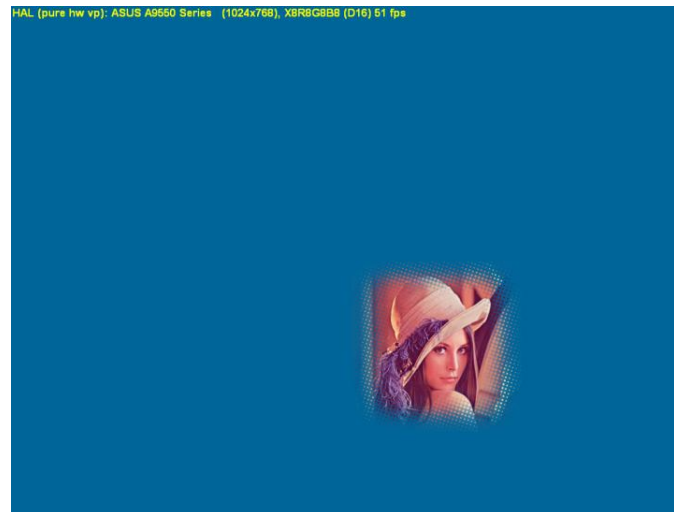
- ◆ STL의 벡터 이용
- ◆ 중복 제거를 위한 키 사용
- ◆ 텍스처를 찾기 위한 Search 알고리즘

- 텍스처 정보 함수

- ◆ 생성 함수
- ◆ 폭, 높이 전달 함수
- ◆ 렌더링 함수

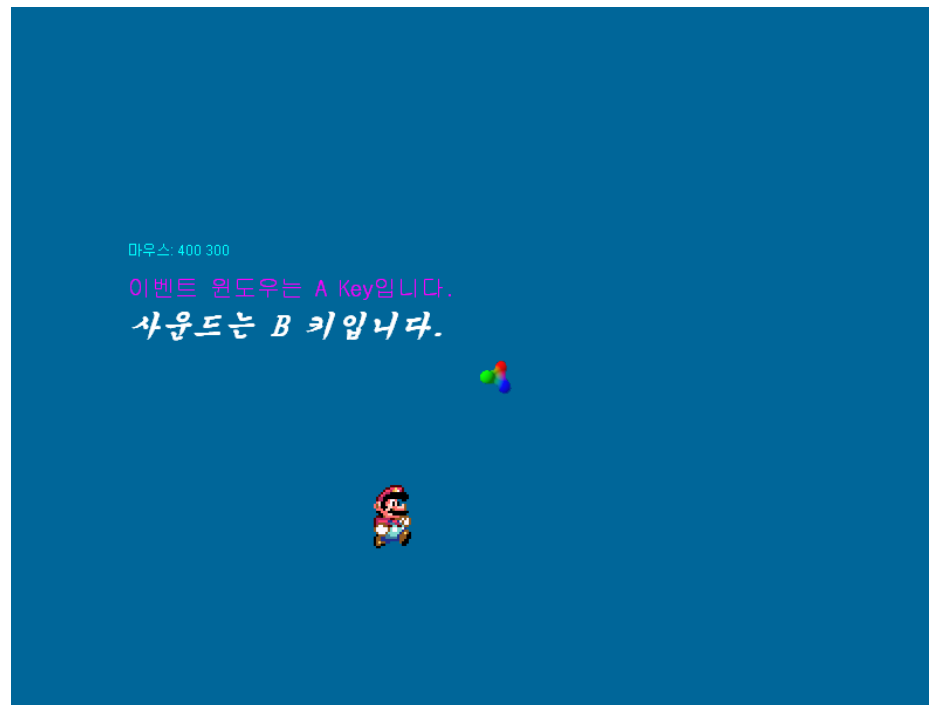
Mcl\_TextureLoad()  
Mcl\_TextureWidth, Height()  
Mcl\_TextureDraw()

- 예제





- 자료 구조는 텍스처와 유사
  - ◆ STL, 키, Search
- 정보 함수들
  - ◆ 생성, 실행 함수들...
- 예제







- 변수 타입
  - ◆ 문자열
  - ◆ double
- 범위
  - ◆ 기본은 전역변수
  - ◆ 지역변수를 사용할 때 변수 앞에 **local** 키워드를 붙임
- 배열
  - ◆ STL의 벡터와 유사 : `g_TxId = {}`
  - ◆ 인덱스 : 1부터 시작
- 함수
  - ◆ `function ~ end`





- 분기

- ◆ if 조건 then 실행 end, if ~ then ~ else ~ end.

- 루프

- ◆ for 시작, 끝, step, do, end

- ◆ while 조건 then 실행 end

- 가변문자 만들기

- ◆ ..연산자: 예) sMsg = "마우스: " .. g\_Mouse[1] .. " "

- 주석

- ◆ "--" 로 시작

- 스크립트에서 스크립트 함수 호출

- ◆ dofile( "스크립트 파일" )





- 루아스크립트는 구조체나 클래스를 지원하지 않음
- 구조체와 비슷한 기능을 하는 테이블제공
  - ◆ 루아 자료구조는 1차원 배열, "Texture={}"
  - ◆ Texture[i].Id=-1 와 같이 멤버처럼 사용하는 변수에 값을 할당하려면 "Texture[i]={}" 먼저 필요

예제)

-루아 테이블(클래스 흉내내기)

**-Texture Class**

**Texture={}**

**-Texture Member Function**

**function Texture.Init(Tx)**

**Tx.iW =Mcl\_TextureWidth(Tx.Id)**

**Tx.iH =Mcl\_TextureHeight(Tx.Id)**

**end**

**function Texture.Draw(Tx, x, y, sX, sY, c)**

**Mcl\_TextureDraw(Tx.Id, 0, 0, Tx.iW, Tx.iH, x, y, sX, sY, c)**

**end**





- 예제 1. 뱅 주사위 게임
- 예제 2. 눈먼 장기
- 예제 3. 병장기
- 예제 4.

