lua编写游戏脚本问题总结

一 生成供lua调用的dll

生成lua可以调用的dll有调用函数的方法和调用类的方法，其中调用函数的方法比较简单

1. 调用函数方法 头文件和源文件如下：

```
// .h
  extern "C"
 {
    //#include "luajit/include/luajit.h"
   #include "lua.h"
   #include "lauxlib.h"
   #include "lualib.h"
 }

 #pragma comment(lib, "lua51.lib")
 extern "C" _declspec(dllexport) int luaopen_AILib(lua_Stat
```

```
// .cpp
int test(lua_State* L)
{
    vector< vector<int> > board = parse_board );/* caffe mod
    std::vector<Legion> legion = statLegionFromBoard board);

    int count = lua_gettop L);
    int legion_count = lua_tonumber L, 1);/* legion count fr
    if (legion_count != legion.size ))
    {
        lua_setglobal L, "arg");
        return 1;
    }
```

```
    lua_createtable(L, legion_count, 0);

    for (int idx = 1; idx <= legion_count; idx++)
    {
        lua_pushnumber(L, idx);
        lua_createtable(L, 0, 5);
        lua_pushnumber(L, legion[idx - 1].id);
        lua_setfield(L, -2, "legion_id");
        lua_pushstring(L, legion[idx - 1].action.c_str());
        lua_setfield(L, -2, "action");
        lua_pushnumber(L, legion[idx - 1].x);
        lua_setfield(L, -2, "x");
        lua_pushnumber(L, legion[idx - 1].y);
        lua_setfield(L, -2, "y");
        lua_pushnumber(L, legion[idx - 1].z);
        lua_setfield(L, -2, "z");
        lua_settable(L, -3);
    }
    lua_setglobal(L, "arg");
    return 1;
}

const struct luaL_Reg TestLib[] =
{
    { "test", test },
    { nullptr, nullptr }
};

int luaopen_AILib(lua_State* L)
{
    luaL_openlib(L, "AILib", TestLib, 0);
    return 1;
}
```

上面的cpp文件中包含了3个函数的实现，`int test()`是dll的功能函数，所有dll需要实现的功能，都在这个函数中定义和调用。值得注意的是，上

述函数的结尾定义了一个arg变量，这个变量是提供给lua script使用的全局变量，里面包含了返回给lua 的所有参数，可以在lua中使用for循环来获取每一个返回值。 `const struct luaL_Reg TestLib[]` 函数是注册函数，这是一个lua table的形式，lua table中存储了一系列的key-value，“test”为key，test为value，test为dll重定义的函数名，即功能函数的函数名，“test”表示在lua script中希望调用这个功能函数的调用名，如何在dll中定义了一系列的功能函数，比如add(),sub(),mul(),div(),那么注册函数的写法如下：

```
float add(){}
float sub(){}
float mul(){}
float div(){}
const struct luaL_Reg TestLib[] =
{
  { "add", add},
  { "sub", sub},
  { "mul", mul},
  { "div", div},
  {nullptr, nullptr}
};
```

注意，上述lua table中的key和value的拼写方法不一定要一致，只要成对出现即可 完成上述代码编写并编译即可生成一个供lua调用的dll，在战意游戏中，lua的依赖项貌似必须是luajit，而lua不行，这可能和游戏中使用的lua版本有关 `int luaopen_AILib(lua_State* L)` 定义的是dll在lua script中的包名，其中luaopen_是固定前缀，AILib是包名，与函数体重“”中的内容保持一致，TestLib是上一个函数注册的lua table名。

定义完上述三个函数，lua调用的dll完成。

二 lua脚本编写

lua脚本的编写也有固定的格式要求

```
-- dll_path为dll的存储路径，luaopen_AILib是dll里面定义的包名，返回
local testlib = package.loadlib(dll_path, "luaopen_AILib")
if(testlib) then
  testlib() //加载dll包
  AILib.test(3) //运行主调函数
  /* 返回一个全局变量arg，后面可以对这个arg进行处理*/
  for i=1,#arg do
    print(arg[i])
```