1. Lua的基础类型

Lua 中有八种基本类型：nil、boolean、number、string、function、userdata、thread和table

1. pairs和ipairs的区别
   1. ipairs 仅遍历表中的数字键值对，且按照键的升序遍历，遇到第一个nil会停止遍历，返回值为索引和值的形式，遍历顺序为有序，按照数字键的升序遍历
   2. pairs 遍历表中的所有键值对，包括数字和非数字键，返回键和值的形式，无序，可能根据哈希值遍历
2. Lua的元表，元方法

元表，即 Lua 中普通 table 的元数据表，而元方法则是元表中定义的普通表的默认行为。Lua 中的每个普通 table 都可为其定义一个元表，用于扩展该普通 table 的行为功能。例如，对于 table 与数值相加的行为，Lua 中是没有定义的，但用户可通过为其指定元表来扩展这种行为；再如，用户访问不存在的 table 元素，Lua 默认返回的是 nil，但用户可能并不知道发生了什么。此时可以通过为该 table 指定元表来扩展该行为：给用户提示信息，并返回用户指定的值。

元表中有两个重要函数：  
 setmetatable(table,metatable)：将 metatable 指定为普通表 table 的元表。  
 getmetatable(table)：获取指定普通表 table 的元表。

\_\_add:+操作。

\_\_sub: - 操作。

\_\_mul: \* 操作。

\_\_div: / 操作。

\_\_mod: % 操作。

\_\_pow: ^ （次方）操作。

\_\_unm: - （取负）操作。

\_\_idiv: // （向下取整除法）操作。

\_\_band: & （按位与）操作。

\_\_bor: | （按位或）操作。

\_\_bxor: ~ （按位异或）操作。

\_\_bnot: ~ （按位非）操作。

\_\_shl: << （左移）操作。

\_\_shr: >> （右移）操作。

\_\_concat: .. （连接）操作。

\_\_len: # （取长度）操作。

\_\_eq: == （等于）操作。

\_\_lt: < （小于）操作。

\_\_le: <= （小于等于）操作。

\_\_index: 索引 table[key]。

\_\_newindex: 索引赋值 table[key] = value

\_\_call: 函数调用操作 func(args)。

1. Lua的table

table 是一个关联数组， 也就是说，这个数组不仅仅以数字做索引，除了 nil 和 NaN 之外的所有 Lua 值 都可以做索引。 （Not a Number 是一个特殊的数字，它用于表示未定义或表示不了的运算结果，比如 0/0。） 表可以是异构的； 也就是说，表内可以包含任何类型的值（ nil 除外）。任何键的值若为 nil 就不会被记入表结构内部。 换言之，对于表内不存在的键，都对应着值 nil 。

表是 Lua 中唯一的数据结构， 它可被用于表示普通数组、序列、符号表、集合、记录、图、树等等。

表、函数、线程、以及完全用户数据在 Lua 中被称为 对象： 变量并不真的持有它们的值，而仅保存了对这些对象的引用。 赋值、参数传递、函数返回，都是针对引用而不是针对值的操作，这些操作均不会做任何形式的隐式拷贝。

1. 进程，线程和协程

进程：

保存在硬盘上的程序运行以后，会在内存空间里形成一个独立的内存体，这个内存体有自己独立的地址空间，有自己的堆，不同进程间可以进行进程间通信，上级挂靠单位是操作系统。一个应用程序相当于一个进程，操作系统会以进程为单位，分配系统资源（CPU 时间片、内存等资源），进程是资源分配的最小单位

线程：

线程从属于进程，也被称为轻量级进程，是程序的实际执行者。线程是操作系统能够进行运算调度的最小单位。它被包含在进程之中，是进程中的实际运作单位。一条线程指的是进程中一个单一顺序的控制流，一个进程中可以并发多个线程，每条一线程并行执行不同的任务。一个线程只有一个进程。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口，但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。线程拥有自己独立的栈和共享的堆，共享堆，不共享栈，线程亦由操作系统调度(标准线程是的)。

协程：

协程是伴随着主线程一起运行的一段程序。协程与协程之间是并行执行，与主线程也是并行执行，同一时间只能执行一个协程提起协程，自然是要想到线程，因为协程的定义就是伴随主线程来运行的。一个线程可以拥有多个协程，协程不是被操作系统内核所管理，而完全是由程序所控制。协程和线程一样共享堆，不共享栈，协程由程序员在协程的代码里显示调度。协成是单线程下由应用程序级别实现的并发。

1. Image和RawImage的区别
   1. Image比RawImage更消耗性能
   2. Image只能使用Sprite属性的图片，RawImage什么样的都可以使用
   3. Image适合放一些有操作的图片，裁剪平铺旋转啥的，针对ImageType属性
   4. RawImage就放单独展示的图片即可，性能比Image好很多