学习ejoy2d——sprite

gaccob

2014 年 3 月 14 日

1. sprite是什么

"sprite是ejoy2d中可以处理的基本图形对象，每个 sprite 都是若干图元以树状组合起来的".

2. sprite属性

结合github上的文档说明，先看一下C kernel中sprite数据结构定义.

```
// file:  lib/spritepack.h
struct sprite_trans {
    struct matrix * mat;
    uint32_t color;
    uint32_t additive;
    int program;
};

// file:  lib/sprite.h
struct sprite {

    // 父节点，与children节点一起，维系了树状结构
    // lua接口: sprite.has_parent(只读), sprite.parent_name(只读)
    struct sprite * parent;

    // 图元类型
    uint16_t type;

    // 唯一id，因为是数组，所以不建议散的太开
    uint16_t id;

    // t.mat 渲染时的变换矩阵，运行期，默认单位矩阵
    // t.color 渲染时的混合颜色，ARGB32，默认为0xFFFFFFFF，作用域是整个
       子树，最常见的是做alpha半透明效果，例如0x80FFFFFF就是50%的半透明
```

```
24      // t.additive 渲染时的叠加颜色, RGB24, 默认为0, 作用域是整个子树
25      // t.program 指定shader
26      // Lua API: sprite.matrix(读写), sprite.color(读写),
            sprite.additive(读写), sprite.program(只读)
27      struct sprite_trans t;
28
29      // 5种基本图元
30      union {
31          struct pack_animation *ani;
32          struct pack_picture *pic;
33          struct pack_polygon *poly;
34          struct pack_label *label;
35          struct pack_pannel *pannel;
36          struct matrix *mat;
37      } s;
38
39      // 只读anchor的特殊属性, 返回上一次这个anchor对象最终渲染的世界矩阵
40      // anchor.visible=false, 当不可显时, 引擎不计算 world matrix
41      // Lua API: sprite.wolrd_matrix(只读)
42      struct matrix mat;
43
44      // 总帧数 与 开始帧数
45      // Lua API: sprite.frame_count(只读)
46      int start_frame;
47      int total_frame;
48
49      // 当前帧号
50      // Lua API: sprite.frame(读写)
51      int frame;
52
53      // 如果设置为false, 则整个子树不显示
54      // Lua API: sprite.visible(读写)
55      bool visible;
56
57      // 对象是否截获 test 调用, 多用于 UI 控制
58      // Lua API: sprite.message(读写)
59      bool message;
60
61      // Lua API: sprite.name(只读)
62      const char *name;
63
64      union {
65          struct sprite * children[1];
66
67          // label的文字
68          // Lua API: sprite.text(读写)
69          const char * text;
70
71          // panel是否有scissor
```

```
72          // Lua API: sprite.scissor(只读)
73          int scissor;
74      } data;
75  };
76
77  // 上面注释中提到的Lua API基本都包含在了getter&setter中
78  // file:  lib/lsprite.c
79  static void
80  lgetter(lua_State *L) {
81          luaL_Reg l[] = {
82                  {"frame", lgetframe},
83                  {"frame_count", lgettotalframe },
84                  {"visible", lgetvisible },
85                  {"name", lgetname },
86                  {"text", lgettext},
87                  {"color", lgetcolor },
88                  {"additive", lgetadditive },
89                  {"message", lgetmessage },
90                  {"matrix", lgetmat },
91                  {"world_matrix", lgetwmat },
92                  {"parent_name", lgetparentname },
93                  {"has_parent", lhasparent },
94                  {NULL, NULL},
95          };
96          luaL_newlib(L,l);
97  }
98
99  static void
100 lsetter(lua_State *L) {
101         luaL_Reg l[] = {
102                 {"frame", lsetframe},
103                 {"action", lsetaction},
104                 {"visible", lsetvisible},
105                 {"matrix" , lsetmat},
106                 {"text", lsettext},
107                 {"color", lsetcolor},
108                 {"additive", lsetadditive },
109                 {"message", lsetmessage },
110                 {"program", lsetprogram },
111                 {"scissor", lsetscissor },
112                 {NULL, NULL},
113         };
114         luaL_newlib(L,l);
115 }
```

上层Lua中, 依据setter&getter接口, 设置sprite的metatable.

```
1  // file:  ejoy2d/sprite.lua
2  function sprite_meta.__index(spr, key)
```

```
3           if method[key] then
4                   return method[key]
5           end
6           local getter = get[key]
7           if getter then
8                   return getter(spr)
9           end
10          local child = fetch(spr, key)
11
12          if child then
13                  return child
14          else
15                  print("Unsupport get " ..  key)
16                  return nil
17          end
18  end
19
20  function sprite_meta.__newindex(spr, key, v)
21          local setter = set[key]
22          if setter then
23                  setter(spr, v)
24                  return
25          end
26          assert(debug.getmetatable(v) == sprite_meta, "Need a
                sprite")
27          method.mount(spr, key, v)
28  end
```

3. sprite方法

看完了C kernel中sprite的数据结构及Lua metatable(属性), 下面来看一下sprite的方法.