链滴

使用 Love2D 写一个贪吃蛇小游戏

作者：ZephyrJung
原文链接：https://ld246.com/article/1472979521095
来源网站：链滴

以下文字均为本人原创，因此有可能说错什么，请不要介意，欢迎指正～

代码由Alexar大大所写（自己实现的那版结构比较乱而且有bug...），群号见下部隐藏区域

####love2d引擎的代码结构

```
function love.load() --资源预加载
end
function love.update(dt) --实时渲染
end
function love.draw() --绘图
end
```

是不是很简单，通过load方法加载资源，或者做一些预处理，通过draw方法绘制想要的图案，再通过pdate不断刷新画面，一个游戏或者动画就形成了。

love2d引擎的游戏，均以main.lua文件作为入口（我就是喜欢这种由确切入口的，有迹可循），而man.lua中，上述三个方法是必须的，其实还有个隐藏的主函数love.run()，用以调用这三个方法。如果什么特殊要求，可以自定义主函数，直接写出就会覆盖（有点像java的构造函数规则）。从官方wiki可以看到run的代码：

```
function love.run()

    if love.math then
        love.math.setRandomSeed(os.time())
    end

    if love.load then love.load(arg) end

    -- We don't want the first frame's dt to include time taken by love.load.
    if love.timer then love.timer.step() end

    local dt = 0

    -- Main loop time.
    while true do
        -- Process events.
        if love.event then
            love.event.pump()
            for name, a,b,c,d,e,f in love.event.poll() do
                if name == "quit" then
                    if not love.quit or not love.quit() then
                        return a
                    end
                end
                love.handlers[name](a,b,c,d,e,f)
            end
        end

        -- Update dt, as we'll be passing it to update
        if love.timer then
            love.timer.step()
            dt = love.timer.getDelta()
        end
```

```
        -- Call update and draw
        if love.update then love.update(dt) end -- will pass 0 if love.timer is disabled

        if love.graphics and love.graphics.isActive() then
            love.graphics.clear(love.graphics.getBackgroundColor())
            love.graphics.origin()
            if love.draw then love.draw() end
            love.graphics.present()
        end

        if love.timer then love.timer.sleep(0.001) end
    end

end
```

通过上述代码（可以不要关心其他方法，单找load，update，draw）可以看到，首先调用的就是loa，然后进入循环体内，不断先后调用update方法，draw方法。

以上可当作love2d的引擎原理，闲话少说，下面进入正题：贪吃蛇

写游戏很能锻炼算法思考，在接触游戏编程之前，我很难理解那些数据结构以及算法到底在哪里用到Java中将大多数数据结构已经封装成类，可以直接调用其提供的各种方法，一般也可以满足需求，那数据结构和算法用在哪里？毕竟，如若没有可以应用的地方，学过就忘（当年纠结了很久终于理解了排序，如今一片茫然，唉……）

我在写贪吃蛇的时候，想了很久如何实现，心中有个猜想但是不太确定，后来群里大神提点了一下肯了我的想法：数组。

整个地图，就是方块矩阵，蛇身也好，苹果也好，墙壁也好，都是矩阵组成，所要控制的，就是如何算出蛇身移动过程中，矩阵坐标的变化，应当考虑一下问题：

- 如何传递蛇头的坐标变化到蛇身的每一节。
- 如何随机生成苹果方块的位置，保证不与蛇身，墙壁重叠
- 蛇吃到苹果后，应该做什么操作（身子加长，苹果刷新，速度加快等）
- 蛇撞到墙壁后如何结束游戏
- 如何区分不动（墙壁），动（蛇，苹果）的绘画

下面上代码，带着这些问题读代码，会更清晰明白些：

首先有如下置顶声明的变量（具体含义，可在下面代码中理解）

```
local updateCD=0.5
local timer=0.5
local dir={x=0,y=1}
local lastDir={x=0,y=0}
local map={}
local food={x=0,y=0}
```

####从load说起

之前提到，load是预加载方法，那么初始化等操作自然就放在这里：

```
function love.load()
    setupMap() --1
```

```
    setupSnake() --2
    drawSnake(true) --3
    newFood() --4
end
```

这里调用了一系列方法，从名字可以看出依次进行了如下操作：

1. 设置地图
2. 设置蛇
3. 画蛇
4. 画苹果

设置地图，就是画出地图边界：

```
local function setupMap()
    map={}
    for x=1,15 do
        map[x]={}
        for y=1,15 do
            if x==1 or x==15 or y==1 or y==15 then
                map[x][y]=true
            else
                map[x][y]=false
            end
        end
    end
end
```

可见，这就是个普通的二位数组赋值方法，用true表示白格，false表示无色（黑色），白格可能代表壁，可能代表蛇身，可能代表苹果，这里就是四周的墙壁。

接下来，设置蛇：

```
local snake={}
local function setupSnake()
    snake={}
    for i=1,5 do
        snake[i]={x=i+5,y=7}
    end
end
```

也是一个简单的赋值函数，不过这里是用lua中table这个数据结构来存储蛇身的位置，以便后面根据身当前位置在地图上画出蛇身。

画蛇，前面已经通过setupSnake设置了蛇身的位置，下面就要在Map上绘蛇了。这里就体现了上面到的动静之分，动态的内容，通过保存其状态值，再通过update不断更新到地图上，而静态的可以接画在地图上。

```
local function drawSnake(toggle)
    for i,v in ipairs(snake) do
        map[v.x][v.y]=toggle
    end
end
```

调用的时候传递了参数true，也就是像填充墙壁一样，填充蛇身。

再下来，画苹果。不难想象，这是用随即函数，生成苹果坐标位置，将其填充到地图上：

```
local function newFood()
    repeat
        food.x= love.math.random(2,14)
        food.y= love.math.random(2,14)
    until check(food.x,food.y)==false
    map[food.x][food.y]=true
end
local function check(x,y)
    return map[x][y]
end
```

检测是否冲突就很简单了，只要看当前map值是否为true即可。上述循环的写法与Java不是很相同，过似乎逻辑是一样（不错，我又晕了，自己体会吧……）

####实时渲染，或者说循环刷新，或者说……

预加载完成了，下面开始写实时渲染逻辑，其实我也不知道这种称呼对不对，通俗点讲就是循环逻辑就好像翻页小人动画一样，这个update就是在不断翻页：

```
function love.update(dt)
    input()
    timer=timer-dt
    if timer<0 then
        timer=updateCD
        updateSnake()
    end
end
local function updateSnake()
    lastDir.x=dir.x
    lastDir.y=dir.y
    local targetX,targetY=snake[1].x+dir.x,snake[1].y+dir.y
    if check(targetX,targetY) then
        if targetX==food.x and targetY==food.y then --eat
            eat()
        else --hit
            gameover()
            return
        end
    end
    drawSnake(false)
    for i=#snake,2,-1 do
        snake[i].x=snake[i-1].x
        snake[i].y=snake[i-1].y
    end
    snake[1].x=targetX
    snake[1].y=targetY
    drawSnake(true)
end
local function input()
    if love.keyboard.isDown("up") then
        if lastDir.y==0 then
            dir.x,dir.y=0,-1
```

```
        end
    elseif love.keyboard.isDown("down") then
        if lastDir.y==0 then
            dir.x,dir.y=0,1
        end
    elseif love.keyboard.isDown("left") then
        if lastDir.x==0 then
            dir.x,dir.y=-1,0
        end
    elseif love.keyboard.isDown("right") then
        if lastDir.x==0 then
            dir.x,dir.y=1,0
        end
    end
end
local function eat()
    table.insert(snake, {x=snake[1].x,y=snake[1].y})
    updateCD=updateCD*0.9
    newFood()
end
local function gameover()
    local title = "Game Over"
    local message = "Your Score is "..tostring(#snake-5)
    local buttons = {"Restart!", "Quit!", escapebutton = 2}
    local pressedbutton = love.window.showMessageBox(title, message, buttons)
    if pressedbutton == 1 then
            setupMap()
        setupSnake()
        drawSnake(true)
        newFood()
        timer=0.5
    elseif pressedbutton == 2 then
        love.event.quit()
    end
end
```

它做了两件事，第一个就是监听输入（上下左右），第二件，就是更新蛇身位置。

须知，update执行的频率十分高，也就是常说的FPS，dt实际上就是每帧间隔时间（也可能是每帧持时间。。。我忘了，先前研究过另一个框架跟这里不同，我现在懒得查验……），为了避免出现疯蛇跑的现象，需要减缓其行动频率，也就是当翻页到了一定时间，才进行一次更新。

updateSnake方法可说是这个游戏的核心算法所在了，主要解决蛇身移动时的坐标计算，如果方向不，蛇身在现有方向前进，如果按下了上下左右键，就应进行相应转向，并且让其无法从反方向走。具我就不解释了，可以试着自己实现下（为了实现贪吃蛇，我也可是进行了一番苦思冥想，自以为用了么新方法，结果与A大这些不谋而合～）

####最后，draw方法：

```
function love.draw()
    drawMap()
end
local function drawMap()
    love.graphics.setColor(255,255,255)
    for x=1,15 do
```

```
    for y=1,15 do
        local how=map[x][y] and "fill" or "line"
        love.graphics.rectangle(how, x*32+100, y*32, 30, 30, 5, 5)
    end
    end
end
```

这就是调用love2d的绘图功能，画出整个内容。这个过程有点像先用白蜡笔在白纸上作图，什么也看到，或者只能看到过模糊的痕迹，最后讲颜料往纸上一泼，一切呈现出来，让人眼前一亮～

大概就是这样了，很简单，不是么？

最后，给出完整代码，只要保存为main.lua，是用配好的环境运行就可以了（放到打赏区咯）：