**local vEnemy = {}**

Tabela de vetores de inimigo.

**local function enemy (xPos, yPos)**

**local x = xPos**

**local y = yPos**

**local mp = function(mx, my, button,istouch)**

**while true do**

**click = false -- Clique fora do inimigo**

**if button == 1 then**

**print("x/y:",mx,xPos,xPos + 210,"/",my,yPos,yPos+136,**

**(mx > xPos) and (mx < xPos + 210) and (my>yPos) and (my<yPos+136))**

**if (mx > xPos) and (mx< xPos +210) and (my>yPos) and (my<yPos+136) then**

**print("clicado")**

**click= true -- Inimigo foi clicado**

**end**

**end**

**mx, my, button,istouch = coroutine.yield (click)**

**end**

**end**

Adquirimos x e y para desenho passados como argumento nas 2 primeiras linhas da função “enemy”. A função mp tem os parâmetros da função comum em love2d “mousepressed” e, depois, o loop infinito necessário para coroutines serem implementadas. O loop começa colocando como falso a variável click. Se houve click com o batão direito, então, button = 1 e nesse caso, precisamos fazer uma verificação se o click foi dentro da imagem com base em 4 condições nessa linha. Mouseclick na coord x foi maior que o x onde a imagem é desenhada ? Mouseclick na coord x foi menor que o local onde a varia´vel é digitada mais seu comprimento (210 pixels)? Se essas 2 forem corretas identificamos que o x está dentro da zona horizontal da imagem, mas resta verificar a zona vertical. Então, verificamos se o click está dentro da zona onde a imagem começa a ser desenhada na vertical e sua largura(136 pixels). Então, diremos que click está true.

Por fim, o click vem como retorno para onde foi chamado, isto é, na parte return dentro da função principal que retrata esse elemento enemy. Lembrando que trazemos também os argumentos do parâmetro dessa função local para, nessa versão, alguma função exterior que no caso é a “mousepressed”.

Existem mensagens de debug para sabermos os valores do click e se ocorreu dentro da imagem; não são necessários para funcionar, mas também não afetam a tela desenhada em no requisito não funcional de robustez.

**local dr = function()**

**while true do**

**love.graphics.setColor(1, 1, 1)**

**enemyImage = love.graphics.newImage("enemy.png")**

**love.graphics.draw(enemyImage, x, y) -- Se por x, y no final ele desenha so 1**

**coroutine.yield()**

**end**

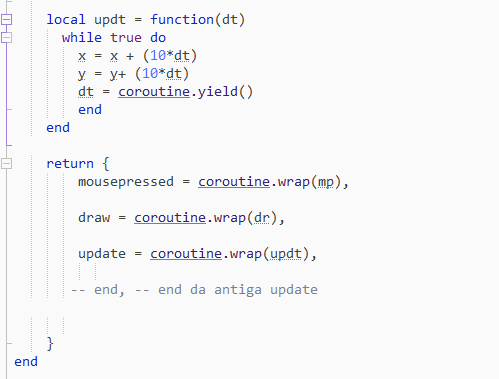
**end**

**A função local dr está atrelada à draw e seus valores vão para draw na parte return da função principal “enemy(xPos,yPos)” que por sua vez são chamados, nessa versão, na função draw do código.**

**O loop infinito é necessário para começar uma courotine, então, os valores não são limpos o que os deixa na memória para a qualquer momento poderem ser alterados porque a função nunca morre graças a esse loop. Além disso, os**

**Os 3 primeiras linhas são:**

* **setColor: em futuras versões colocará desenhos como formatos geométricos como brancos. Essa versão é facultativo a existência.**
* **enemyImage = love.grap… : Carrega em enemyImage a imagem de um inimigo.**
* **love.graphics.draw( … : desenha a imagem do inimigo em x e y que são inicializadas com os valores vindos da função principal “enemy” - lembra que salvamos xPos e yPos como x e y locais ?**
* **Por fim, o coroutine.yield() parará a função em memória, o que é bom quando implementarmos mais detalhes como movimento, por exemplo. Toda essa função quando executada, pelo menos deixará o desenho na tela.**

****

**enemy(xPos, yPos)** -> essa função tem as funções que atendem a um elemento inimigo. encapsuladas O return possui funções que quando chamadas ativarão suas respectivas funções. A update não possui uso nessa versão do código. O draw e mousepressed possuem courotines para as funções desejadas que são chamadas a partir do nome no lugar do parâmetro em “coroutine.wrap(<PARAMETRO>)”.

A função update retorna a função da coroutine da função local “updt”, conforme vemos no parâmetro dela. Ela será executada fora da função enemy na update.

updt é uma função local e tem como parâmetro dt. O loop essencial para coroutine começa. “x” e “y” serão acrescentados em 10 \* o tempo que passou, llembrando que encapsulamos x e y dentro da função enemy nas 2 primeiras linhas da mesma. Por fim, “dt= coroutine.yield()” está pausando a função até que a mesma seja chamada novamente trazendo de volta para “update” na parte de returns o valor do parâmetro de updt e nenhum outro retorno porque os parênteses após yield estão vazios..

**function love.load()**

**love.window.setTitle("HonorNDuty")**

**table.insert(vEnemy, enemy(50,50) )**

**table.insert(vEnemy, enemy(250,250) )**

**table.insert(vEnemy, enemy(50,350) )**

**table.insert(vEnemy, enemy(350,35) )**

**table.insert(vEnemy, enemy(350,450) )**

**table.insert(vEnemy, enemy(450,350) )**

**end**

**LOVE.LOAD** -> Coloca título na janela contendo string com o nome do game e inserts na tabela “vEnemy” com a função de criação de inimigos no segundo parâmetro. Os números após “enemy(“ correspondem às coordenadas em que serão desenhados na tela.

**-- Função que lida com cliques do mouse**

**function love.mousepressed(x, y, button, istouch)**

**-- Itera sobre a lista de inimigos e verifica se algum foi clicado**

**local i = 1**

**while true do**

**if i > #vEnemy then break end**

**status = vEnemy[i].mousepressed(x, y, button, istouch)**

**if status then**

**print("Remove")**

**table.remove(vEnemy, i) -- Remove o inimigo clicado da lista**

**else**

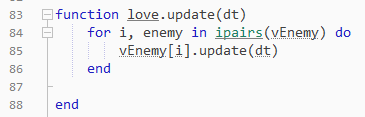
**i= i + 1**

**end**

**end**

**end**

**LOVE.MOUSEPRESSED(X,Y,BUTTON.ISTOUCH)** -> Ela está associada aos cliques do mouse. “X” e “Y” recebem as coordenadas do mouse quando um clique ocorre, “button” adquire qual botão foi e “istouch” é booleana para a tecnologia de tela touch. “i” é uma variável local para contador. O loop infinito começa até que o contador seja maior que a quantidade de elementos na tabela. O status será booleano e se a função do elemento em pesquisa pelo for retornar true teremos 1. Então, o elemento é retirado da tabela através do parâmetro da tabela em si e o contador, o que fará a imagem do inimigo desaparecer. Apenas haverá incremento no contador se não achamos nenhum elemento do vetor de inimigos que corresponda ao click em si. A impressão perto de remove apenas indica que entrou ali para caso de debug.

****

LOVE.UPDATE -> A função “for” por id(i) e sua chave que é um elemento “enemy varreu todos os enemys da tabela e executará a update levando o parâmetro dt.

**function love.draw()**

**for i, enemy in ipairs(vEnemy) do**

**enemy.draw()**

**end**

**end**

LOVE.DRAWN -> O “for” ativará a função draw encapsulada em cada elemento da tabela “vEnemy”. O id “i” não está sendo usado, mas o elemento atrelado a essa chave da tabela está sendo usado, o qual é “enemy”.