EDUSCRATCH

PROGRAMANDO COM O

ARRASTAR SPRITES

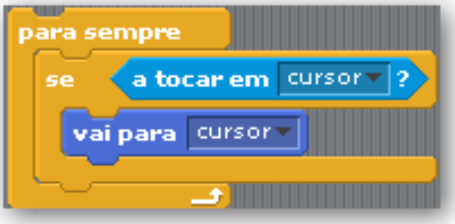
Técnica de controlo do arrastamento de sprites

(Out-2010)

No ecrã de desenvolvimento, os sprites podem ser arrastados livremente; mas no ecrã de apresentação ou 'online', só se conseguem arrastar os sprites se tiverem o cadeado aberto ou se existirem comandos com essa finalidade.

O método do "cadeado aberto" é accionado com um clique sobre o cadeado existente, a meio do ecrã, no topo. É expedito mas não permite que o programa 'saiba' para onde o sprite foi arrastado nem mesmo se activa o evento "quando o sprite é clicado".

Assim, assume alguma importância o desenvolvimento de um bloco de comandos para permitir e controlar o arrastamento dos sprites. É o que eu vou tentar explicar neste documento, tendo como referência o meu projecto "Estados Brasileiros", (http://scratch.mit.edu/projects/ffred/652250 e http://kids.sapo.pt/scratch/projects/ffred/1201), em que o utilizador tem de arrastar as figuras dos Estados para os seus lugares geográficos, num mapa do Brasil.



Desde logo se detecta a necessidade do sprite ficar sempre alerta para a eventualidade de ser tocado pelo rato, evento que é coberto pelos comandos à direita.

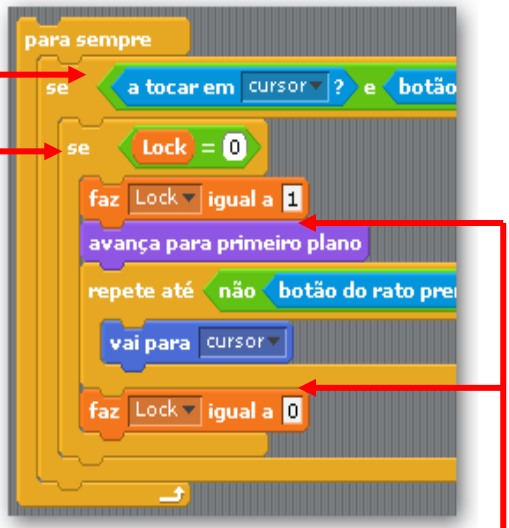
Mas o que parece resolver o problema acaba por criar vários problemas e o menor não será, certamente, o de largar o sprite do rato, já que eles se movimentarão sempre juntos, para sempre. (Vá fazendo e experimentando).

Uma condição composta pode resolver este problema:

Agora, o sprite só vai seguir o rato enquanto o botão esquerdo deste estiver premido. Ao largar o botão, larga-se também o sprite que, por força do ciclo "para sempre", fica à espera que o rato lhe seja clicado em cima novamente.

Parece perfeito. Imagine-se, porém, que existem vários sprites no ecrã: quando se arrasta um deles, pode sempre tocar-se noutro que, naturalmente, se agarrará também ao rato, a par com o primeiro e sempre junto com ele mesmo quando se largar o botão. Na verdade, precisamos aqui de um mecanismo que iniba os outros sprites de se agarrarem ao rato quando este já estiver a arrastar um.

Então, criamos uma variável, (por exemplo: **Lock**) que é inicializada a **0** e que toma o valor **1** quando um sprite está a ser arrastado, só voltando a **0** quando ele é largado.



Analisemos este bloco: só quando o rato com o botão premido e tocar no sprite, é que se avança; e testa-se o valor de "Lock". Se não for zero, é porque um sprite qualquer está a ser arrastado neste momento, logo, este sprite ignora a passagem do rato sobre si e volta ao ciclo "para sempre", aguardando outra oportunidade. Se for zero, é porque o rato está desimpedido e podemos continuar com os comandos. (É algo como a luz no tejadilho dos táxis: acesa (0) podemos usá-lo; apagada (1) já está ocupado).

A primeira coisa que se faz é mudar o valor de "Lock" para 1, impedindo outros sprites de se aproximarem. Segue-se um grupo de comandos, (que já analisaremos) e, no fim, torna a fazer-se Lock=0, senão nunca mais se conseguirá arrastar qualquer sprite, (nem mesmo este, na próxima oportunidade).

Os comandos no meio do bloco podem ser os mais diversos, conforme o que queremos fazer com o sprite arrastado; mas é de bom tom que o façamos passar para primeiro plano, (senão arrastá-lo-íamos sob outros que, eventualmente tivessem maior precedência); e o ciclo "repete até que o botão do rato não esteja premido" serve para que a acção não passe de imediato ao comando "faz Lock=0", estragando-nos o esquema todo.

Este é um esquema genérico para arrastar sprites controlando bem os acontecimentos. Mas, no caso concreto do projecto "Estados Brasileiros", eu pretendia que os sprites fossem arrastados para locais específicos e, assim, acrescentei comandos para testar se o sprite era largado no sítio certo ou não. Também não queria conceder segunda oportunidade, isto é, quando o sprite fosse largado uma vez, não deveria voltar a ser arrastado. Tendo isto em mente, vamos ver como foi possível programá-lo...

Comecei por criar mais uma variável de nome “Fixo” e âmbito local, isto é, apenas conhecida de um sprite. (Claro que cada sprite terá a sua variável exclusiva); e convencionei que esta variável seria 0 até ser arrastada e largada, altura em que passaria a ser 1.

Depois, acrescentei mais uma condição para que o sprite pudesse ser arrastado. Vamos ver como ficou o “se”:

Agora, para que um sprite seja arrastado, é preciso que o rato o toque, com o botão premido, que não haja outro sprite a ser arrastado, (Lock=0) e que este sprite nunca tenha sido mexido (Fixo=0). Assim que o utilizador o largar, “Fixo” passará a ser 1 e o sprite fica “***inarrastável***”.

Analisemos a parte nova, (a amarelo), da figura junta: Se o sprite foi largado no lugar certo, (já veremos qual a condição composta que aqui foi introduzida no vermelho), toca um som, (é um adorno) e “vai para x,y”, em que (x,y) é o lugar exacto onde o sprite pertence.

Mau! Se eu testo se o sprite foi largado no seu lugar, para que é preciso mandá-lo para lá? É que raramente o utilizador acerta com o local exacto, pelo que é preciso dar uma distância de tolerância para aceitar que ele acertou. Depois, é de bom tom que se afinem as suas coordenadas.

Já veremos como é que esta tolerância se deve reflectir na condição colocada na zona vermelha.

Continuando: se o utilizador acertou, soma-se ainda 1 ponto à variável contadora dos pontos. Se acaso não acertou, toca outro som, (mais um adorno), coloca-se o sprite no sítio certo mas, desta vez, com um “deslize” em vez de um “vai para” e, naturalmente, tira-se 1 aos pontos.

Logo a seguir, faz-se “Fixo=1” para que este sprite não volte a ser mexido. O comando fica fora do “se” porque tanto faz que acerte como não, o sprite é para imobilizar.

Depois do “Lock=0” que já estudámos, acrescentei um “pára este bloco” para terminar o “para sempre”, uma vez que deixa de ser necessário ocupar a memória com o ciclo de um sprite que não vai voltar a ser mexido.

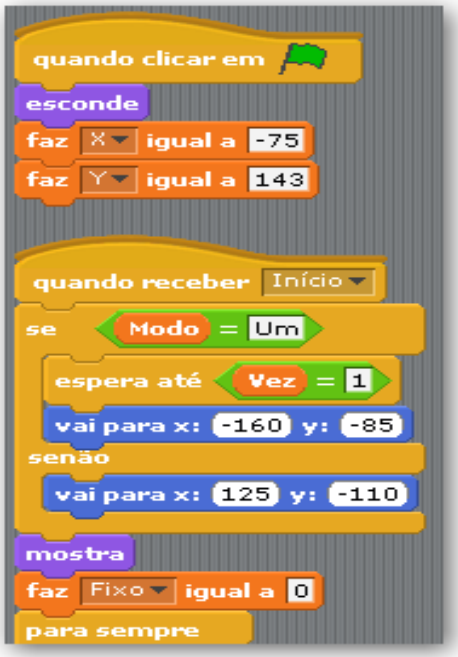
É altura de analisar a condição que vai julgar se o sprite está no lugar ou não:

As variáveis de sistema “posição x” e “posição y” guardam em permanência as coordenadas da posição do sprite. Admitindo que o ponto (x, y) é o local exacto de colocação, podemos ver que se subtraíram as coordenadas homónimas, reduzindo-se as diferenças aos seus valores absolutos e que se aliaram duas condições em que se testa se as distâncias em abcissas e em ordenadas são menores do que um determinado valor que foi atribuído à variável “Tol”.

Vamos lá mais devagar: para avaliar a proximidade, nem foi preciso calcular a distância mas apenas o valor absoluto das diferenças entre coordenadas. A variável “Tol” justifica-se pois, sem primeiro experimentar, o programador não tem a noção da tolerância a usar para decidir com justiça se o sprite foi largado perto ou longe do lugar. Então o melhor é criar uma variável com um valor provisório e usar a variável na construção do projecto. Se a prática vier a aconselhar a utilização de outra tolerância, basta alterar o valor que se lhe atribuiu sem necessidade de andar pelos sprites todos a emendar os valores.

Por razões muito parecidas é que usei as variáveis X e Y em vez dos valores absolutos. Quando um dos sprites fica com as suas instruções completas, basta copiá-las para os outros sprites, emendando apenas o que for diferente, como é o caso das coordenadas. Ora as coordenadas surgem em vários locais pelo que haveria muito que emendar. Usando variáveis as locais X e Y pode copiar-se à vontade pois só é preciso emendar no início, como veremos a seguir.

Para quem queira entreter-se a analisar o resto dos comandos do projecto, vou deixar aqui mais umas dicas. Os comandos iniciais são os seguintes, em cada sprite:



...aqui liga-se ao que já conhecemos.

- Há um primeiro bloco que define o valor das variáveis X e Y para este sprite, tal como acima referi. Estas duas são variáveis locais e vale a pena falar um pouco sobre o que são variáveis locais e variáveis globais, (mais à frente).
- O primeiro bloco "esconde" o sprite porque o controlo do início do programa está no Palco, como veremos adiante; e o controlo só passa aos sprites com o anúncio "Início"
- Este anúncio activa o segundo bloco onde, à cabeça, se vai posicionar o sprite com comandos "vai para x y". Como se vê, há duas posições possíveis, conforme o "Modo" tem ou não tem o valor "Um". Isto explica-se porque eu quis fazer duas modalidades de jogo: uma mais fácil em que todos os sprites estão visíveis e espalhados pelo ecrã; e outra mais difícil em que aparece um sprite de cada vez. (Com todos os sprites no ecrã, pode pegar-se no que der mais jeito para o colocar, ajudando assim a definir a posição de outros; mas com um de cada vez, tem que se colocar o que aparecer, nem que seja um do meio).
- Verifica-se que, no modo "Um", este sprite fica à espera que a variável "Vez" assuma o valor 1, (este foi um número que eu dei aos sprites para que cada um só responda quando a variável "Vez" tiver o seu número; mas não é um ordinal, já que os sprites vão ser sorteados, como veremos mais à frente. No caso de Modo=Um, este sprite vai para (-160, -85) que é onde todos hão-de aparecer, um de cada vez. No caso de Modo não ser Um, este sprite vai para (125, -110) que é onde não choca com nenhum dos outros, quando aparecem todos no ecrã.
- Segue-se o comando "mostra" para que o sprite apareça, (em qualquer dos modos) e a inicialização da "Fixo" que é a tal variável local que é posta a 1 para que o sprite não possa tornar a ser arrastado. (Ver início da página 2).

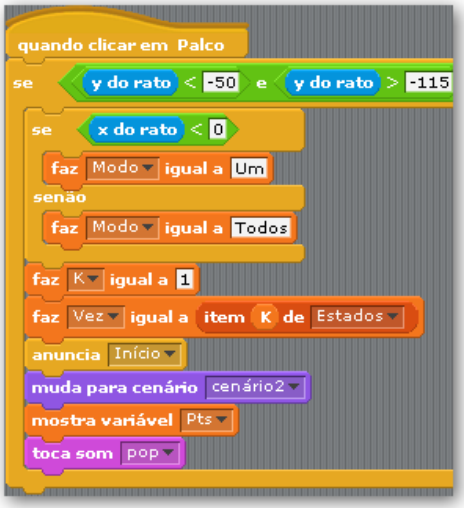
Terminada esta explicação, vamos espreitar ao Palco o que acontece quando se clica na bandeira verde:

- A variável que conta os pontos é escondida para que não estorve o "cenário1" inicial onde é explicada a finalidade do jogo; e são inicializadas as variáveis que, recorde-se:
  - Lock=0 indica que nenhum sprite está a ser arrastado
  - Modo=Todos é para virem todos os sprites de uma vez
  - Tol=10 é a tolerância de colocação de um sprite
  - Vez=0 é a variável que indica que sprite deve vir para o ecrã, no modo "um Estado de cada vez"
  - Pts=0 é o contador de pontos obtidos
  - K=0 é um índice que, mais à frente, vai servir para contar quantos sprites já foram colocados. Por agora vai servir para encher aleatoriamente uma lista com os números que eu atribuí aos sprites.
- Falemos de listas: uma lista é um conjunto de variáveis, todas com o mesmo nome, que se distinguem pelo índice, que mais não é do que o seu numero de ordem dentro da lista. Neste bloco, começo por apagar todos os valores da lista "Estados" e, depois, incrementando K de 1 até 26, vou inserir aleatoriamente esses valores numa lista. É claro que vão ficar todos baralhados e é isso que se pretende.

Vejamos: Quando K=1, o número 1 é colocado no primeiro lugar da lista, pois não há ainda números lá colocados; mas quando K=2, o número 2 pode ser colocado antes ou depois do 1; e quando K=3, este

pode ser colocado no início, no meio ou no fim da lista. Se for no início, o 1 deixa de ser o primeiro. O que é facto é que, ao fim de ter colocado 26 números na lista "Estados", eles ficam todos baralhados. Então, se eu começar a chamar, por ordem, o primeiro da lista, depois o segundo e por aí adiante até ao 26º, conseguirei que a variável "Vez" assuma esses 26 valores aleatoriamente e sem repetir nenhum, (que é o que interessa para chamar os 26 sprites).

O segundo bloco do Palco é activado pelo clique do utilizador, numa das duas áreas em que deve escolher o modo de jogar. Se o clique for dado em qualquer lugar do Palco acima de y=-50, é ignorado; e abaixo deste valor, se for dado à esquerda do ecrã. faz Modo=Um, se for dado à direita faz Modo=Todos.



A partir daqui, "K" assume o valor 1 e "Vez" é feita igual ao primeiro valor da lista, (imagine-se que é o 16). Quando é emitido o anúncio "Início" todos os sprites ficam à espera de ver sair o seu número em "Vez"; mas só o que estiver à espera de "Vez=16" é que vem para o ecrã, (se o modo escolhido tiver sido o "Um", porque se foi o "Todos", vêm todos os sprites para o ecrã, de uma vez.

A finalizar, o cenário muda para um mapa do Brasil só com os contornos, a variável "Pts" é exibida e toca um som qualquer, para alegrar.

Por aqui se vê como é chamado o primeiro sprite; mas, e os seguintes, como serão chamados? Voltamos aos blocos dos sprites, onde ainda não falámos dos dois comandos que chamam o sprite seguinte, até que todos os 26 são chamados.

Se bem se lembram, havia uma altura em que Fixo=1, Lock=0 e, depois, se parava o bloco para poupar memória. A verdade é que existem lá mais dois comandos: um que altera "K" para "K+1", ou seja, o item seguinte da lista. Outro que coloca esse item na variável "Vez". Sabe-se que há um sprite que tem estado à espera que "Vez" assuma" o seu valor, pois é esse sprite que avança; e, quando acabar, volta a aumentar "K" para que "Vez" assuma novo valor.

Até que "K=27", depois de ter despachado o 26º item da lista. Nesta altura, acorda o sprite "FIM" que tem instruções para esperar até que "K>26".

O "FIM" avalia o valor da variável "Pts" e emite uma mensagem a propósito, acompanhada de um pedacinho de música, num rectângulo que cresce desde o tamanho 10% até ao 160%.

Só falta falar de variáveis Locais e variáveis Globais. As primeiras só são conhecidas de um sprite. Desta forma, cada sprite pode ter as suas variáveis exclusivas com os mesmos nomes das que existem noutros sprites, pois só ele as conhece. É o caso das variáveis "Fixo", "X" e "Y.

As variáveis Globais não admitem repetição de nomes e são conhecidas e manipuladas por qualquer sprite. É o caso das "K", "Lock", "Modo", "Pts", "Tol" e "Vez".

Define-se o âmbito de uma variável, quando é criada:

Frederico