EDUSCRATCH

PROGRAMANDO COM O

COMANDO E CONTROLO

Controlo do fluxo de um programa

(Out-2010)

C3 é um conceito que se aplica à programação em Scratch como, de resto, a todas as actividades humanas que envolvam ordens de execução, (comandos), avaliação de resultados (controlo) e troca de informação (comunicações). Estas capacidades estão reunidas nos comandos da categoria "Controlo" do Scratch e permitem que as acções se executem pela ordem mais adequada à resolução do problema.

Basicamente, os comandos de um programa são executados, sequencialmente, pela ordem em que estão colocados; mas há figuras que introduzem alterações a esta regra: o "se", a "escolha múltipla", o "ciclo", a "subrotina", a "execução simultânea" e a "recursividade". O Scratch só não realiza a "escolha múltipla" nem a "recursividade"pelo que anda bem perto das outras linguagens.

O "se" estabelece uma condição e conforme ela seja falsa ou verdadeira, assim o fluxo de execução continua por um ou por outro ramal; é como uma bifurcação na estrada.

A "Escolha Múltipla(Case)" é uma bifurcação em estrela, em que o caminho é escolhido de acordo com o valor de uma variável, possibilitando assim escolher entre várias opções. É como uma estrada, com faixas para lentos, rápidos, ciclistas peões e Bus.

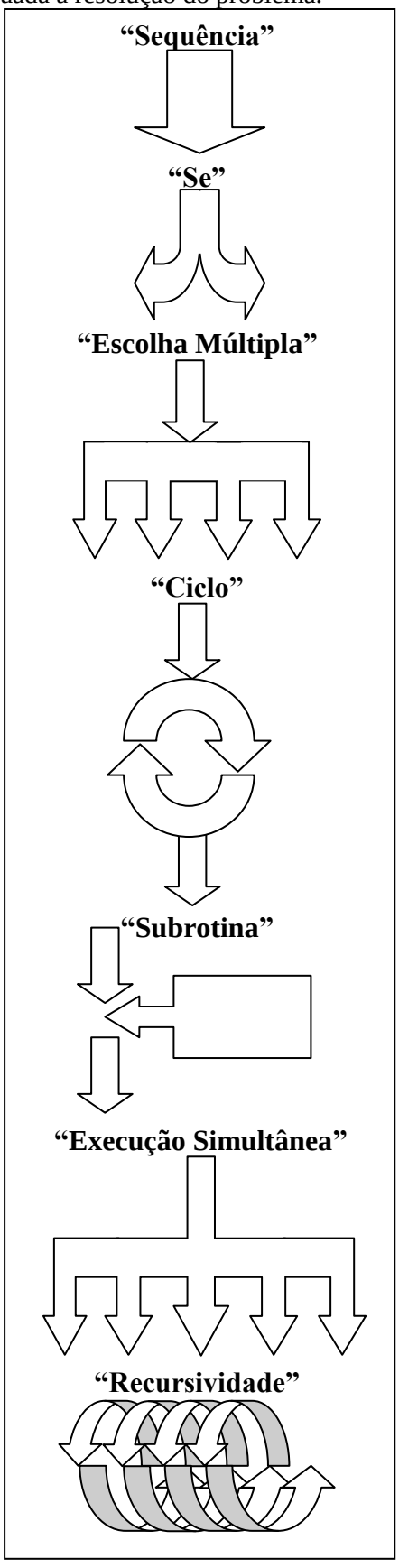
O "ciclo" interrompe a sequência, repete um determinado número de vezes um conjunto de comandos, retomando depois a execução sequencial no comando a seguir ao conjunto repetido: é como uma rotunda.

A "subrotina" interrompe a sequência, executa um conjunto de comandos, (separado do programa principal), e volta a engrenar no comando sequencial a seguir ao da interrupção: é como uma ida às boxes, durante uma corrida.

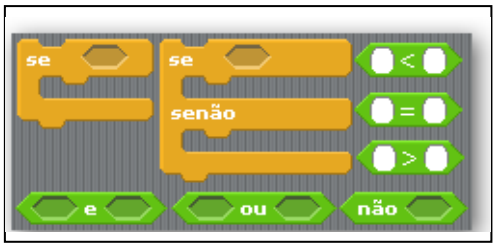
A "execução simultânea" permite que os comandos de vários blocos sejam executados, senão em simultâneo, dando, pelo menos essa sensação. Na verdade, a Unidade Central de Processamento do computador vai atendendo um comando de cada vez, ora de um bloco ora de outro, sem favoritismo, como a ponte 25 de Abril vai escoando as intermináveis filas do seu garrafão.

A "recursividade" é a capacidade de repetir um conjunto de comandos, avançando sempre, como se fosse um parafuso que anda um passo de rosca por cada rotação. Na recursividade, um bloco de comandos, auto-copia-se e todas as suas cópias vão ficando em memória como tijolos em cima uns dos outros, pelo que, quando a acção tem um fim, as cópias desmontam-se de memória pela ordem inversa de que se foram montando. No Scratch, sempre que um bloco de comandos se chama a si próprio para se auto-repetir, recursivamente, é apenas um começar de novo, do princípio, e só existe uma cópia em memória.

Tudo simples como com as notas musicais. A dificuldade, tal como na música, reside nas infinitas possibilidades de composição que estas figuras oferecem, bem como nas próprias variantes de cada uma das figuras. Teremos oportunidade de ver alguns exemplos, já a seguir.



"Se"



O Scratch tem dois tipos de "se", (os de cor sépia na figura da esquerda). O mais simples enquadra comandos a executar se a condição for verdade; e que não serão executados se a condição for falsa.

O outro pode enquadrar um conjunto de comandos para o caso da condição ser verdadeira e outro conjunto para o caso dela ser falsa. As condições são as de cor verde, bicudas, adequadas aos espaços a preencher nos "se". São operações lógicas que podem ser compostas entre si de maneiras variadas.

Por exemplo, neste primeiro bloco, a variável "tempo" é incrementada a cada segundo e, assim que atinge o valor 60, torna a condição verdadeira. Desta forma, o comando "pára este bloco" é executado; e é uma boa maneira de interromper um ciclo infinito ao fim de um minuto.

Mas podia haver necessidade de conjugar o tempo com o valor de outra variável, por exemplo, "pontos". Neste caso, teria de se compor a condição, como no exemplo seguinte:

Ora isto levanta o problema de saber qual das duas condições simples é que tornou falsa a condição composta. Terá sido a variável "tempo" que não chegou a 60, ou é "pontos que já passou de 9?

Se isso for importante, temos que reformular o "se", talvez, desta maneira, em que: se o tempo não atingir 60 nem vale a pena ver mais nada; mas se o minuto se completou, vamos ver se os pontos estão abaixo de 10 e, neste caso, paramos o bloco, como já tínhamos decidido; porém, se há, pelo menos, 10 pontos, concedemos mais um minuto, reiniciando a variável tempo a zero.

Eis como podemos ir combinando os "se" entre si, conforme as conveniências do programa; e facilmente se intui que não há limite para as combinações de "se" e de condições.

"Escolha Múltipla"

É nisso que se baseia a simulação de "escolha múltipla" que aqui se repruduz. Como o Scratch não a permite, encaixam-se os "se" dentro uns dos outros e o conjunto funciona como se fosse uma "escolha múltipla". Admitindo que "pontos" só pode assumir os valores do intervalo [1, 5], testa-se se é igual a 1 e, se não for, testa-se se é igual a 2. Se não for, testa-se se é igual a 3 e ainda se é igual a 4; mas já não será preciso testar se é igual a 5, porque aqui chegando, já só pode ser 5.

"Ciclo"

Há no Scratch quatro formatos para executar ciclos. O ciclo "para sempre" é infinito e, logo, não proporciona um meio "legal" de o parar, como acontece com o "repete N vezes" que, após ter executado os comandos do seu interior por "N" vezes, continua a execução no comando seguinte, já fora do ciclo.

Repare-se como este ciclo termina com um encaixe para o comando seguinte, coisa que não existe no "para sempre", já que é suposto que nunca pára.

Contudo, é possível incluir dentro dele uma condição, como a que vimos acima, para "parar o bloco" quando um condição for verdadeira. Podemos usar este tipo de recurso em qualquer dos ciclos; mas é particularmente útil nestes ciclos "para sempre".

Uma variante do "para sempre" simples é o que usa uma condição para saber quando deve executar ou não os comandos que tem no seu interior. Esta variante também fica carregada em memória para sempre, embora só execute quando a condição for verdadeira.

Podemos dizer que o simples está sempre a executar e o da condição está sempre à espreita para ver quando a condição passa a ser verdade; e sempre que isso aconteça, executa. Quando a condição deixa de ser verdade, fica outra vez à espera da próxima vez em que ela volte a ser verdadeira.

Nos ciclos do tipo “repete” também existe uma condição, quer seja o número de repetições que é testado em cada volta para não ser ultrapassado, quer seja uma condição propriamente dita.

O curioso é que, o “repete até” que uma condição seja verdadeira pode não executar vez alguma, desde que a condição seja verdade logo no início; mas se só ficar verdadeira no meio dos comandos do ciclo, são sempre cumpridos todos os comandos até ao fim, pois só na volta é que a condição é avaliada.

Uma utilização comum do repete com uma condição, é o que se apresenta à direita, em que se inicializa uma variável que vai ser incrementada dentro do ciclo até que a condição fique verdadeira e o ciclo termine.

Claro que neste exemplo, em que pontos vai incrementar 10 vezes 2 até igualar 20, mais valia usar um “repete 10 vezes”; mas há casos em que não sabemos à partida de quanto a variável vai crescer, até porque pode ser o resultado de uma operação aritmética. Nestes casos faz todo o sentido deixar as coisas correr até que a variável de controlo (é assim que se chama) assuma um valor que achamos ser o limite.

Os ciclos são particuarmente úteis quando se usam variáveis lista, já que podemos usar a variável de controlo como índice da posição de um item na lista. É o que se vê no exemplo à direita.

Mas o grande poder dos ciclos aumenta quanto os aninhamos dentro uns dos outros. Imagine-se que é preciso dizer a tabuada. “A” começa em 0 e entra num ciclo em que é logo incrementado em 1. A seguir é a vez de “B” ser feito igual a 0 e entrar num ciclo onde logo é incrementado em 1.

Sai então a mensagem com o resultado da multiplicação de “A” por “B” que, nesta altura corresponderá a “1 x 1”.

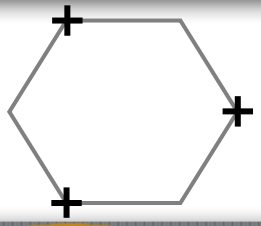
Após um segundo de espera, repete-se o ciclo mais interno e “B” passa a 2, mostrando-se o resultado de “1 x 2”; e assim continua durante as 10 voltas do ciclo interno, em que “B” assume valoes de 1 a 10.

Nesta altura, termina o ciclo interno e cai-se no externo que incrementa “A” de 1 para 2 e torna a inicializar “B” para mais 10 voltas e para fazer sair a tabuada dos 2.

Claro que isto só termina quando o ciclo exterior der 10 voltas; e a última mensagem a sair será a do resultado de 10 x 10

“Subrotina”

As subrotinas são úteis para quando se tem um conjunto de comandos que se repete em vários locais do programa e, por vezes até em vários programas. Em vez de estar a repetir esse conjunto de comandos, basta apenas chamá-los para que venha dar a sua contribuição onde for preciso.



Imagine-se que queremos desenhar uma cruz em três pontos do ecrã equidistantes entre si e todos a uma dada distância de um outro definido pelo utilizador ou sorteado pelo Scratch.

Da análise se depreende que vai ser preciso desenhar um hexágono, cujo centro e raio são dados. Seria fácil desenhar um sprite com as cruzes em três vertices do hexágono e aplicá-lo com o comando “carimba” no local pedido com um factor de escala que lhe adequasse o raio e até com o requinte de uma rotação que orientasse os três vértices; mas vamos escolher a maneira difícil, porque precisamos de exemplificar o uso de subrotinas: faremos isto com os comandos da “Caneta”.

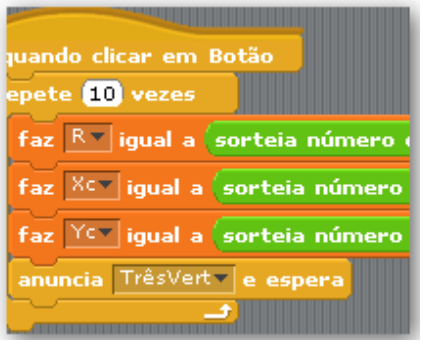
Primeiro, fazemos a subrotina que há-de traçar uma cruz quando for chamada por um comando “anuncia cruz”. Admite-se que (Xv,Yv) são as coordenadas do vértice onde a cruz vai ser

colocada e, como se vê no último bloco da página anterior, serão feitos dois traços cruzados, com o comprimento de 10, centrados em (Xv,Yv). (Antes da subrotina "cruz" ser chamada, terá sido definida a espessura e a cor do traço, características estas que podem variar de cruz para cruz ou mesmo de hexágono para hexágono.)

Depois, fazemos o bloco que há-de calcular (Xv,Yv) e chamar "cruz", três vezes. Este bloco é activado com um "anuncia TrêsVert" (que algo há-de emitir) e prepara as coordenadas (Xv,Yv) que são usadas pela subrotina "cruz".

Repare-se que passa o controlo da execução para a subrotina e fica à espera que ela termine para calcular o ponto seguinte. Usando o comando "anuncia" sem "espera" daria resultados muito estranhos, porque ainda a cruz estava a ser desenhada e já havia novo par de coordenadas e nova chamada da "cruz".

(Não calculei o Xv do último ponto porque é igual ao do 2º.)



Finalmente, construímos o bloco que vai chamar "TrêsVert", passando-lhe os valores sorteados do Raio e do Centro e repetindo o processo 10 vezes.

Depois de chamar a subrotina, espera que esta conclua a sua tarefa, isto é, que mande desenhar as cruzes e espere que a "cruz" termine.

(Estes três blocos podem estar todos no mesmo sprite ou em sprites separados, uma vez que as variáveis são globais, ou seja, são conhecidas de todos os sprites.)

Durante este exemplo, foram usadas cinco variáveis e algumas constantes; mas tudo podia ser baseado em variáveis, desde que, na altura de chamar as subrotinas, sejam definidos os seus valores. Assim, a rotina "cruz" serviria para todas as cruzes possíveis e não apenas para cruzes de 10 de envergadura; e a "TrêsVert" serviria para qualquer triângulo.

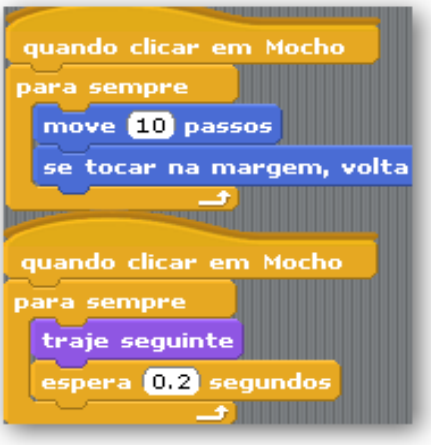
Este é o conceito básico de "Objecto": algo que tem em si todos os recursos e comandos para realizar uma tarefa, cujas variáveis se pré-definem. (Os sprites são objectos, em linguagem informática)

Uma janela Windows é um objecto usado por muito programas que lhe definem a posição, as dimensões, a sobreposição e muitas outras características. É que, depois de um objeto estar definido, podemos usá-lo onde for preciso sem nos importarmos como são feitos por dentro. Parece uma caixa negra a que fornecemos alguns parâmetros variáveis para obter um resultado especializado.

Também merece referência o uso dos comandos "anuncia" e "quando receber" porque preenchem a necessidade de comunicação entre os sprites que concretiza no Scratch a teoria C3.

"Execução Simultânea"

Esta capacidade do Scratch é muito importante, embora deva ser usada com parcimónia porque o lançamento de vários blocos (subrotinas) em simultâneo pode produzir resultados inesperados, sobretudo se usarem os mesmo recursos ou dependerem umas das outras.



Mas há ocasiões em que são indispensáveis, como no exemplo junto em que uma ave se desloca por todo o ecrã, batendo as asas, por alternância entre os seus vários trajes.

No primeiro bloco, a ave desloca-se 10 passos, sem parar e, no segundo bloco, também sem parar, alterna entre os seus trajes que provocam a animação do bater das asas.

Frederico