Estudar Na Prática

Voltar

Anterior Próximo

Como mexer no Scratch?

Podemos escrever programas com os blocos de construção que acabamos de descobrir:

- funções
- condições
- Expressões booleanas
- rotações

E descobriremos recursos adicionais, incluindo:

- variáveis
- tópicos
- eventos ...

Antes de aprendermos a usar uma linguagem de programação baseada em texto chamada C, usaremos uma linguagem de programação gráfica chamada Scratch , onde arrastaremos e soltaremos blocos que contêm instruções.

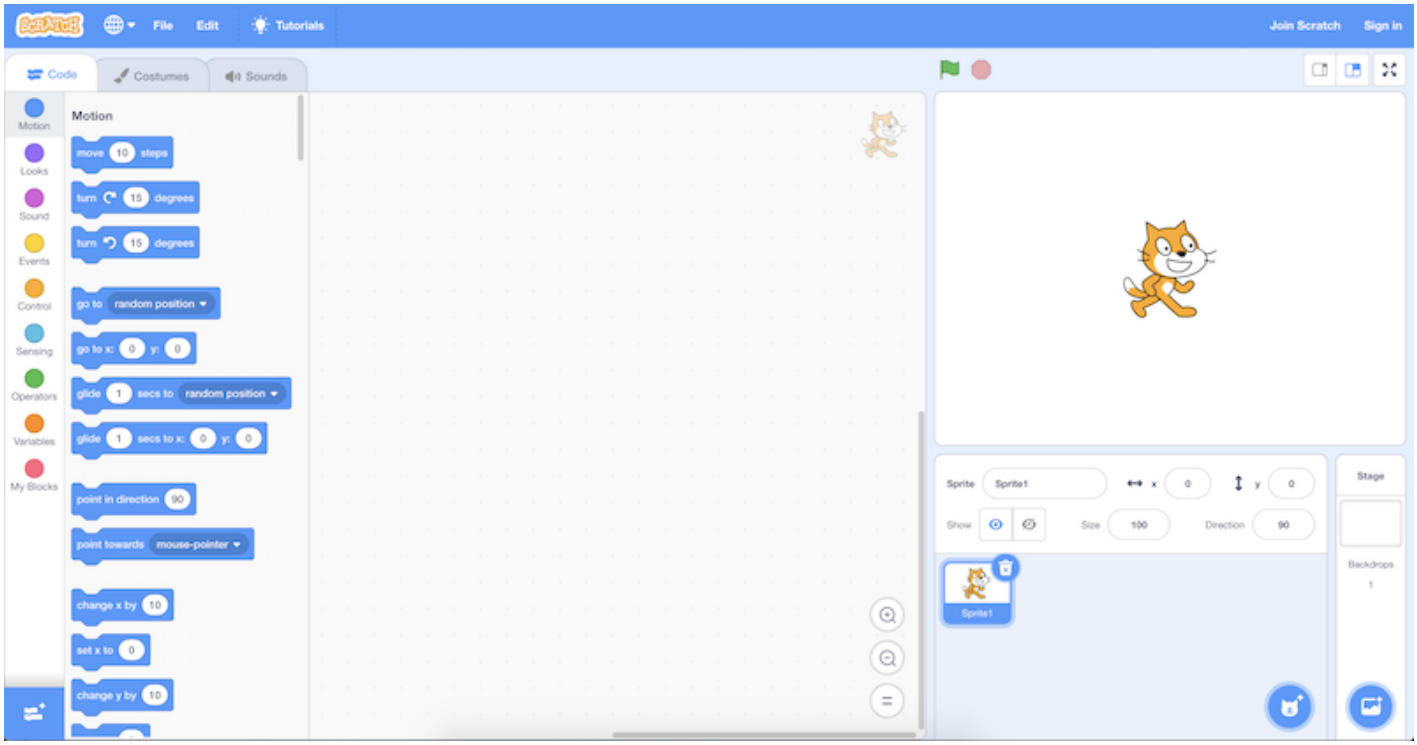
Um programa simples em C que imprime “olá, mundo”, ficaria assim:

```
#include <stdio.h>

int main(void)
{
    printf("oi, mundo\n");
}
```

- Há muitos símbolos e sintaxe, ou seja, o arranjo desses símbolos, que teríamos que descobrir.

O ambiente de programação do Scratch é um pouco mais amigável:

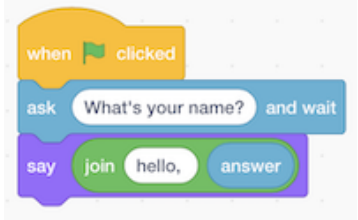


- No canto superior direito, temos um “palco” que será mostrado pelo nosso programa, onde podemos adicionar ou alterar planos de fundo, personagens (chamados de sprites no Scratch) e muito mais.
- À esquerda, temos peças de quebra-cabeça que representam funções ou variáveis, ou outros conceitos, que podemos arrastar e soltar em nossa área de instrução no centro.
- No canto inferior direito, podemos adicionar mais personagens para nosso programa usar.

Podemos arrastar alguns blocos para fazer o Scratch dizer “olá, mundo”. Ao adicionar o bloco “**green flag clicked**” (“quando a bandeira verde é clicada”) refere-se ao início do nosso programa (já que há uma bandeira verde acima do palco que podemos usar para iniciá-lo), e abaixo dela encaixamos um bloco “**say**"(“dizer”) e digitamos “hello, world” ("olá mundo"). E podemos descobrir o que esses blocos fazem explorando a interface e experimentando.

Também podemos arrastar o bloco “**ask _ and wait**” (“perguntar _ e esperar”), com uma pergunta como “what’s your name” (“qual é o seu nome?”), e combiná-lo com um bloco “**say**” para a resposta. O bloco “**answer**” (“responder”) é uma variável, ou valor, que armazena o que o usuário do programa digita, e podemos colocá-lo em um bloco “**say**” arrastando e soltando também.

Mas não pausamos depois de dizer “Olá” com o primeiro bloco, então podemos usar o bloco “**join**” (“juntar”) para combinar duas frases para que nosso gato possa dizer “olá, David”:

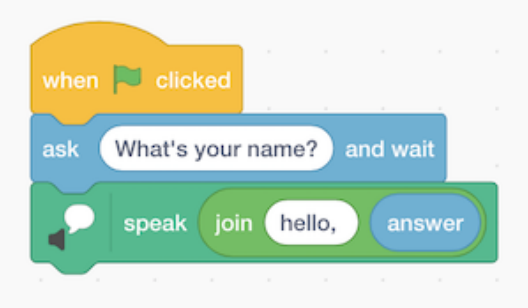


Quando tentamos aninhar blocos ou colocá-los uns dentro dos outros, o Scratch nos ajudará a expandir os locais onde eles podem ser usados. Na verdade, o bloco "**say**" em si é como um algoritmo, onde fornecemos um input de "olá, mundo" e ele produziu a output de Scratch (o gato) "dizendo" essa frase.

O bloco "**ask**" também recebe um input (a pergunta que queremos fazer) e produz um output do bloco "answer".

Podemos então usar o bloco “**answer**” junto com nosso próprio texto, “hello,“, como duas entradas para o algoritmo de junção... o output do qual podemos passar como input para o bloco "**say**".

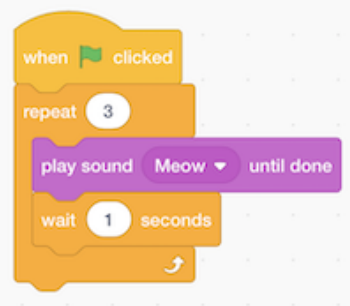
No canto inferior esquerdo da tela, vemos um ícone para extensões, e um deles é chamado "**text-to-speech**”. Depois de adicioná-lo, podemos usar o bloco "**say**" para ouvir nosso gato falar:



A extensão "**text-to-speech**”, graças à nuvem, ou servidores de computador na internet, está convertendo nosso texto em áudio. Podemos tentar fazer o gato dizer miau:

```
when green flag clicked
play sound: Meow until done
wait 1 seconds
play sound: Meow until done
wait 1 seconds
```

Podemos dizer miau três vezes, mas agora estamos repetindo blocos indefinidamente. Vamos usar um loop ou um bloco de “**repeat**” (“repetição”):



Agora nosso programa atinge os mesmos resultados, mas com menos blocos. Podemos considerar que ele tem um design melhor: se há algo que queremos mudar, só precisaríamos mudar em um lugar ao invés de três.

Podemos fazer com que o gato aponte para o mouse e se mova em direção a ele:

```
when green flag clicked
forever
   point towards mouse-pointer
   move 1 steps
```

Experimentamos a extensão da caneta, usando o bloco “**pen down**” ("caneta para baixo") com uma condição:

```
when green flag clicked
forever
   go to mouse-pointer
   if mouse down? then
      pen down
```

```
   else
      pen up
```

Aqui, movemos o gato até o ponteiro do mouse, e se o mouse for clicado, ou para baixo, colocamos o "**pen down**", que desenha. Caso contrário, colocamos a caneta para cima. Repetimos isso muito rapidamente, uma e outra vez, e então produzimos o efeito de desenhar sempre que mantemos o mouse pressionado.

Scratch também tem diferentes fantasias, ou imagens, que podemos usar para nossos personagens. Faremos um programa que pode contar:

```
when green flag clicked
set counter to 1
forever
     say counter for 1 seconds
     change counter by 1
```

Aqui, "counter" é uma variável, cujo valor podemos definir, usar e alterar. Vemos mais alguns programas, como o salto , em que o gato se move para frente e para trás na tela para sempre, girando sempre que estivermos na borda da tela.

Podemos melhorar a animação fazendo com que o gato mude para uma roupa diferente a cada 10 passos no bounce1. Agora, quando clicamos na bandeira verde para executar nosso programa, vemos o gato alternar o movimento de suas pernas.

Podemos até gravar nossos próprios sons com o microfone de nosso computador e reproduzi-los em nosso programa.

Para construir programas cada vez mais complexos, começamos com cada um desses recursos mais simples e os colocamos em camadas um dentro do outro. Também podemos fazer o Scratch miar se tocarmos com o ponteiro do mouse, no pet0.

Na bark, não temos um, mas dois programas no mesmo projeto Scratch. Ambos os programas serão executados ao mesmo tempo depois que a bandeira verde for clicada. Um deles tocará um som de leão-marinho se a variável **silenciada** for configurada como **falsa**, e o outro configurará a variável silenciada de **verdadeiro** para **falso** ou de **falso** para **verdadeiro**, se a tecla de espaço for pressionada.

Outra extensão olha para o vídeo conforme capturado pela webcam do nosso computador e reproduz o som de miado se o vídeo tiver movimento acima de algum limite.

Com vários sprites ou personagens, podemos ter diferentes conjuntos de blocos para cada um deles:

```
when green flag clicked
forever
if key [space] pressed? then
     say Marco! for 2 seconds
     broadcast event
```

Para um fantoche, temos esses blocos que dizem "Marco!" E, em seguida, um bloco de "**broadcast**" ("transmitir"). Este "evento" é usado para nossos dois sprites se comunicarem, por exemplo enviando uma mensagem nos bastidores. Portanto, o nosso outro fantoche pode simplesmente esperar por este "evento" para dizer "Polo!":

```
when I receive event
say Polo! for 2 seconds
```

Também podemos usar a extensão “Translate” para dizer algo em outros idiomas:

```
when green flag clicked
ask Qual é seu nome? and wait
say translate join [olá] answer to English
```

Aqui, o resultado do bloco “**join**” é usado como entrada para o bloco “**translate**”, cujo resultado é passado como entrada para o bloco “**say**”.

Agora que sabemos algumas noções básicas, podemos pensar sobre o design ou a qualidade de nossos programas. Por exemplo, podemos querer que o gato mia três vezes com o bloco "**repeat**"(repetir):

```
when green flag clicked
repeat 3
 play sound Meow until done
     wait 1 seconds
```

Podemos usar abstração, o que simplifica um conceito mais complexo. Neste caso, podemos definir nosso próprio bloco “miau” no Scratch e reutilizá-lo em outro lugar em nosso programa, como visto em miau3 . A vantagem é que não precisamos saber como o miado é implementado ou escrito em código, mas apenas usá-lo em nosso programa, tornando-o mais legível.

Podemos até definir um bloco com uma entrada em meow4 , onde temos um bloco que faz o gato miar um certo número de vezes. Agora podemos reutilizar esse bloco em nosso programa para miar qualquer número de vezes, da mesma forma como podemos usar os blocos “traduzir” ou “falar”, sem saber os detalhes de implementação ou como o bloco realmente funciona.

Vamos dar uma olhada em mais algumas demos, incluindo Gingerbread tales remix e Oscartime, que combinam loops, condições e movimento para criar um jogo interativo. Oscartime foi na verdade feito por David muitos anos atrás, e ele começou adicionando um sprite, então um recurso de cada vez, e assim por diante, até que eles acabassem formando um programa mais complicado.

Um ex-aluno, Andrew, criou o Raining Men. Embora Andrew tenha acabado não seguindo a ciência da computação como profissão, as habilidades de resolução de problemas, algoritmos e ideias que aprenderemos no curso são aplicáveis em todos os lugares.

Até a próxima vez!

Anterior Próximo

Em caso de dúvida, envie email para relacionamento@estudar.org.br

Plataforma de ensino por