

Mini-curso *Scratch*



**Desenvolvimento do Raciocínio
Lógico no Ensino Fundamental e Médio**

Organização: Vinícius Borges Martins
Fevereiro de 2017.

DESENVOLVIMENTO DO RACIOCÍNIO LÓGICO NO ENSINO FUNDAMENTAL E MÉDIO

Participantes:

- Coordenadores:
Diana Adamatti
Karina Machado
Fernanda Mota

- Bolsistas:
Janaína Adolfo da Silva
Vinícius Borges Martins

Sumário

1.	Introdução	1
2.	Instalando a ferramenta.....	2
3.	Conhecendo a ferramenta	3
3.1	Ambiente de Programação Offline (versão antiga)	3
3.2	Ambiente de Programação Online e Offline (nova versão).....	7
4.	Movimento.....	15
5.	Eventos.....	18
6.	Controle.....	19
7.	Aparência	21
8.	Operadores.....	23
9.	Variáveis	25
10.	Caneta	27
11.	Sensores	28
12.	Exercícios.....	31

1. Introdução

A ferramenta *Scratch* foi criada pelo Grupo do MIT (*Massachusetts Institute of Technology* - ‘Instituto Tecnológico de Massachusetts’), em colaboração com o grupo de Yasmin Gafai na UCLA (*University of California, Los Angeles* - ‘Universidade de Califórnia, Los Angeles) e se baseia nas ideias de *Logo* [6], mas substitui o código digitado por uma abordagem de “arrastar-soltar” inspirado em *LogoBlocks* [7] e em *Etoys* [8]. A ferramenta enfatiza a manipulação de mídias e apoia as atividades de programação que despertam interesse em jovens, tais como a criação de histórias animadas, jogos e apresentações interativas.

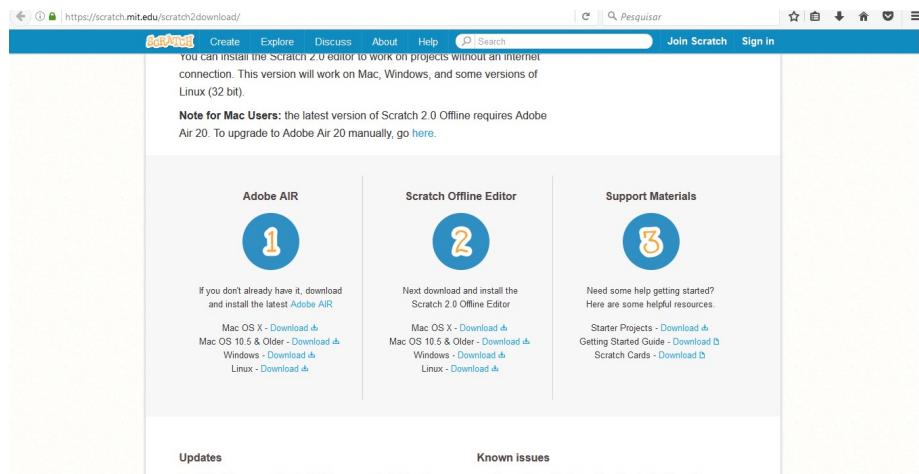
O desenvolvimento de projetos na ferramenta é fácil, pois fornece um ambiente que possibilita a construção de projetos com o uso de “blocos” de comando, tais como: estruturas de repetição, condicionais, comunicações, sons e variáveis, trabalhando com a introdução de conceitos de lógica que na maioria das vezes não são apresentados nas séries finais do ensino fundamental e ensino médio. Todos esses conceitos são vistos de forma implícita na ferramenta ajudando os alunos ingressantes nos cursos de computação. O *Scratch* tem sido utilizado no auxílio do aprendizado de Java em um curso introdutório de ciência da computação na *Universidade de Harvard* [12].

O *Scratch* é diferente dos ambientes tradicionais usados para introduzir ideias de ciência da computação, pois seu ambiente de programação de “arrastar-soltar” incentiva a experimentação, elimina problemas de sintaxe e permite que os alunos se concentrem na resolução de problemas e no projeto do algoritmo. Sua ênfase em gráficos, animação, som e interação com o usuário permite que os alunos trabalhem em problemas que envolvam o seu interesse com a criação das mídias disponíveis [4, 6, 10, 11, 12].

Os blocos de comando permitem que os alunos interajam com a ferramenta quase totalmente com o mouse, pois seus blocos de programação têm formas que se encaixam sintaticamente. Estes blocos dão “vida” aos *sprites*(personagens / objetos) e, se colocados num *palco*, tem como resultado final a animação, jogo ou arte interativa do aluno.

2. Instalando a ferramenta

Para realizar o *download* do *Scratch offline* o aluno deverá entrar no site <http://scratch.mit.edu/download> e seguir o passo a passo a seguir. Para o uso da ferramenta *online* basta acessar http://scratch.mit.edu/projects/editor/?tip_bar=home.



Passo 1: Acesse o site.

Passo 2: Faça *download* e instale o *Adobe AIR* clicando no *link* ao lado do nome ao seu sistema operacional (Se já possuir o *software* no computador, pule esse passo).

Passo 3: Faça *download* do executável do *Scratch* e o instale. O *download* é feito do mesmo jeito que o *download* do *Adobe AIR*, mas o *link* pertencendo à etapa 2 da figura.

Pronto! A última versão do *Scratch* está instalada no seu computador.

Obs.: Com a instalação mostrada acima, será instalada a versão *Scratch2*, tendo esta o *layout* diferente de sua versão anterior e idêntica à versão *online*.

3. Conhecendo a ferramenta

Como citado anteriormente, o *Scratch* está disponível em duas versões: *online* e *offline*. Ambas serão mostradas a seguir (inclusive a versão antiga da *offline*).

3.1 Ambiente de Programação Offline (versão antiga)

O ambiente de programação *offline* do *Scratch* (Figura 2) possui os seguintes comandos:



- **Ações do Script**



→ : Começa a executar o programa que o aluno fez.

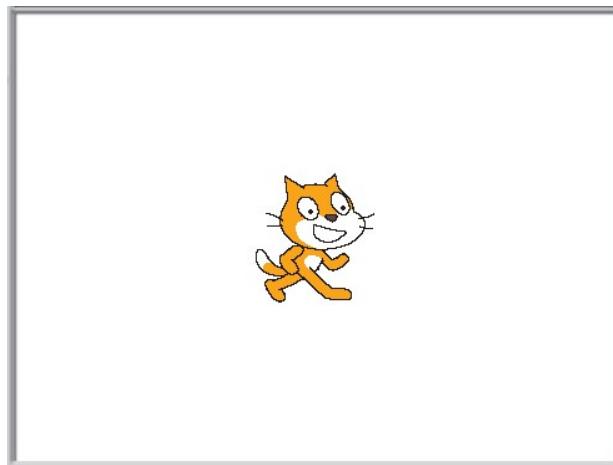
→ : Finaliza o programa que o aluno fez.

- **Edição de Objeto**



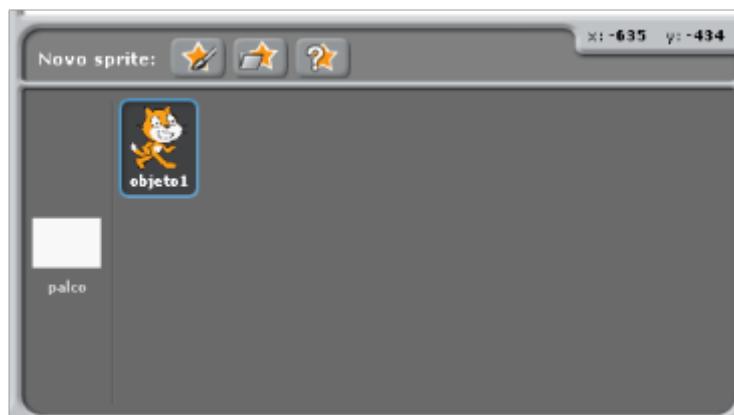
- : Duplica um objeto. → : Aumenta o objeto desejado.
- : Exclui um objeto. → : Diminui o objeto desejado.

- **Palco**



Onde aparecem os objetos visíveis e a imagem de fundo do programa no momento.

- **Objetos Utilizados**

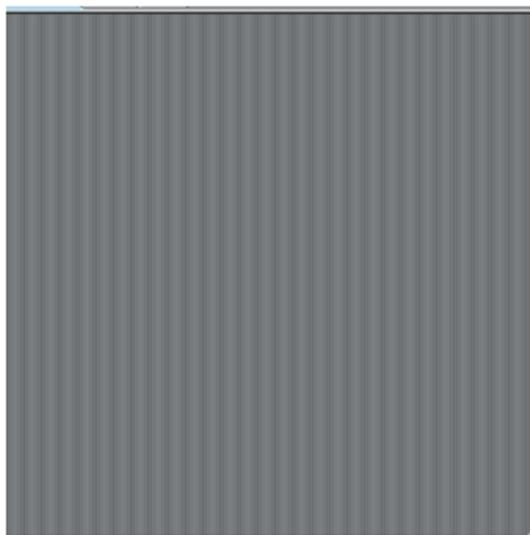


- : Desenhar ou importar um objeto.
- : Escolher um objeto surpresa.
- : Escolher um objeto do arquivo.
- : Coordenadas cartesianas (x e y) do mouse.

- **Opções do Objeto**



- : Opções de giro do objeto. De cima para baixo: gira em todos os sentidos, gira somente para um lado e para outro (horizontal) e fica virado para somente um lado.
- : Define o ângulo de movimento do objeto.
- : Mostra as coordenadas cartesianas do objeto e o ângulo que foi definido para o movimento.
- : Da esquerda para direita:
- Comandos: Área aonde o aluno vai por os blocos e montá-los em forma de algoritmo.



- Trajes: Onde personaliza os trajes dos objetos (adiciona, exclui, renomeia..).



- Sons: Reservado para configurações de sons do programa (gravar, importar...).

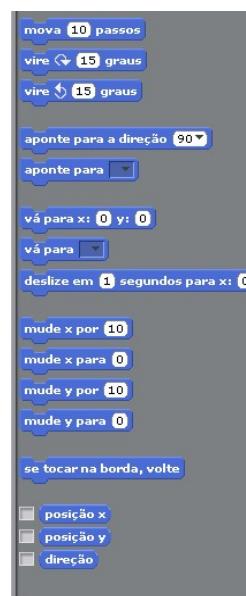


- **Categorias de Comando**



Categorias dos blocos de comando. Cada uma apresenta um conjunto de blocos.

- **Blocos de Comando**



Os blocos são usados para fazer o programa. Devido à forma de como o *Scratch* foi criado, eles se encaixam sintaticamente, ou seja, encaixarão somente se for sintaticamente correto.

- **Comandos da Ferramenta**



Barra superior do *Scratch*. Dividida em:

→ : Selecionar o idioma.

→ : Salvar projeto.

→ : Compartilhar projeto.

→ **Arquivo**: Subdividido em:

- Novo arquivo (cria um novo projeto);
- Salvar (salva o projeto);
- Abrir (abre um projeto);
- Salvar como (salva o projeto com outro nome);
- Importar projeto (importa um projeto que já existe para o projeto atual).
- Exportar projeto (exporta o projeto);
- Notas do projeto (insere notas/observações sobre o projeto);
- Sair (Fecha a janela do *Scratch*).

→ **Editar**: Consiste em:

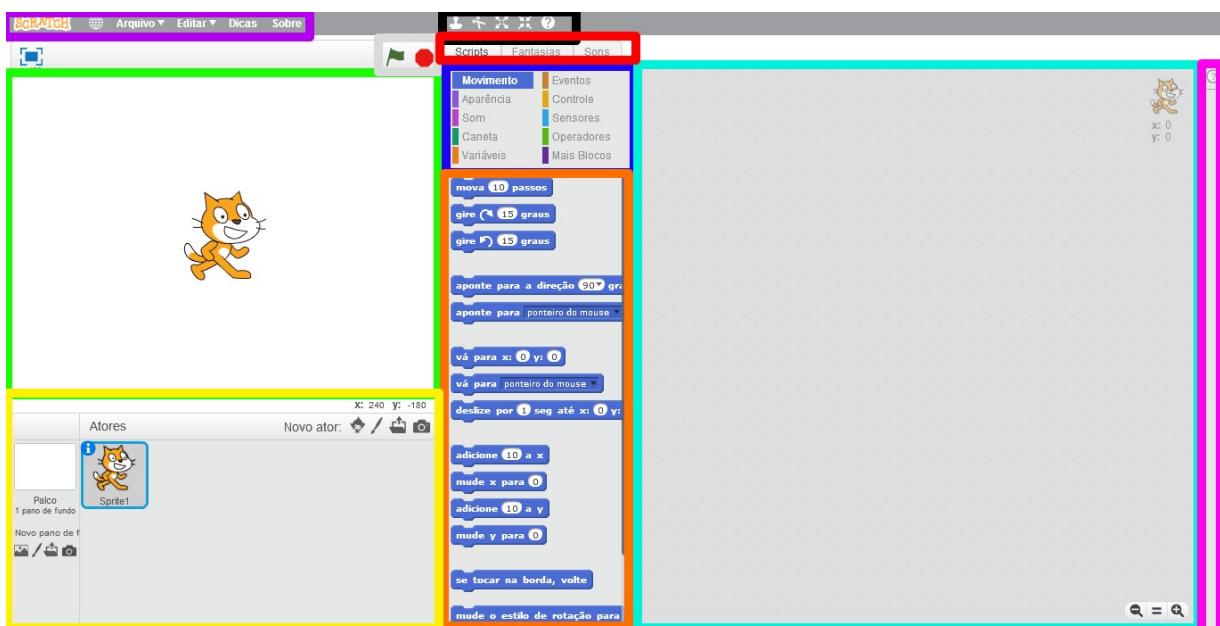
- Desfazer apagar (recupera o último *Sprite* ou bloco apagado);
- Iniciar passo a passo (mostra o passo a passo da execução do programa);
- Marcar passo simples (muda a velocidade do passo a passo);
- Comprimir sons (comprime os sons, reduzindo o tamanho do projeto);
- Exibir blocos motor (ativa essa categoria de blocos).

→ **Compartilhar**: Compartilha o projeto na internet.

→ **Ajuda**: Ajuda da ferramenta.

3.2 Ambiente de Programação Online e Offline (nova versão)

A seguir são demonstrados os ambientes *online* e *off-line*–atualizado- do *Scratch*:





- **Comandos da Ferramenta**



A barra superior se divide em:

- : Selecionar idioma;
- **Arquivo ▾** :
 - Novo: abre um projeto em branco;
 - Abrir: abre um projeto salvo anteriormente;
 - Salvar: salva o projeto;
 - Salvar como: salva o projeto com outro formato;
 - Gravar um projeto de vídeo: grava um vídeo com a animação do seu projeto e salva no seu computador. Só podem ser gravados 60 segundos de vídeo por vez.
 - Compartilhar no site: exporta o projeto para o site do Scratch;
 - Verificar atualizações: verifica se há alguma atualização mais recente que a sua;
 - Sair: sai da ferramenta.
- **Editar ▾** :
 - Recuperar: cancela a exclusão de um bloco ou *sprite*;
 - Disposição com palco pequeno: diminui o tamanho do palco para ter uma área de edição de *script* maior;
 - Modo turbo: acelera a reprodução do *script*.
- **Dicas** : Faz deslizar a barra lateral de ajuda;
- **Sobre** : Abre uma página web com as informações do software.

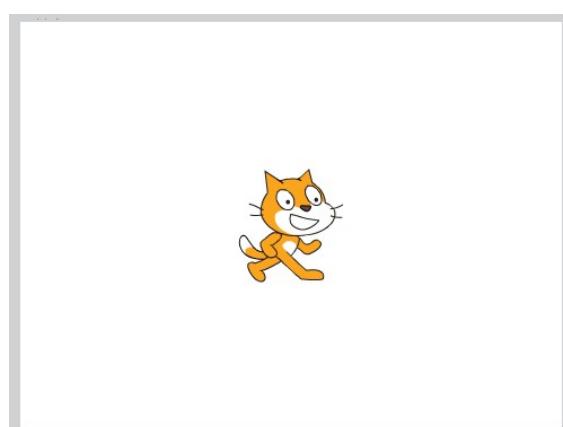
- **Ações do Script**



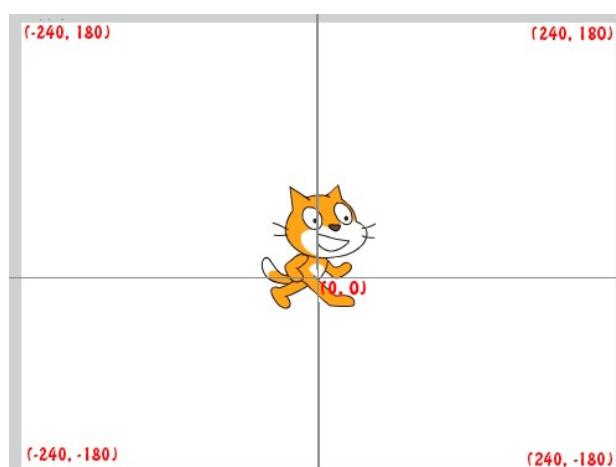
→ : começa a executar o *script*;

→ : para a execução.

- **Palco**

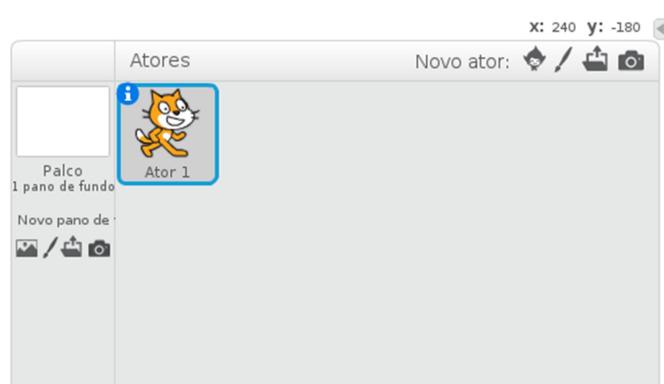


Onde aparecem os objetos visíveis e a imagem de fundo do programa no momento. O centro do palco é a coordenada (0, 0). Os extremos são mostrados abaixo:



- **Objetos Utilizados**

Área que aparece todos os objetos incorporados ao projeto. É possível encontrar as seguintes funções, também:



→ Para inserir um novo ator (*sprite*):

- : escolhe da biblioteca interna do *Scratch*;

- : desenha um novo personagem;
- : importa do computador;
- : tira foto para usar como ator.

→ Para definir um novo plano de fundo:

- : escolhe um *background* da biblioteca do *Scratch*;
- : desenha seu próprio plano de fundo;
- : importa uma imagem dos documentos do computador;
- : utiliza a câmera para tirar uma foto de um *background*.

→ `x: 162 y: -180` : mostra as coordenadas cartesianas do mouse;

- **Edição de Objeto**

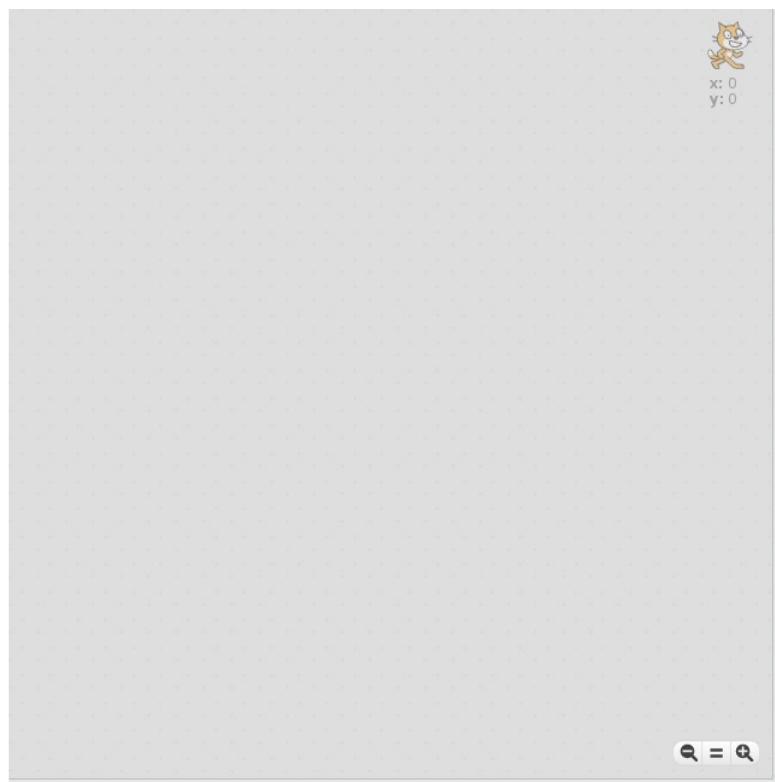


- : duplica o objeto;
- : exclui um objeto;
- : aumenta um objeto;
- : diminui um objeto;
- : abre a aba de ajuda.

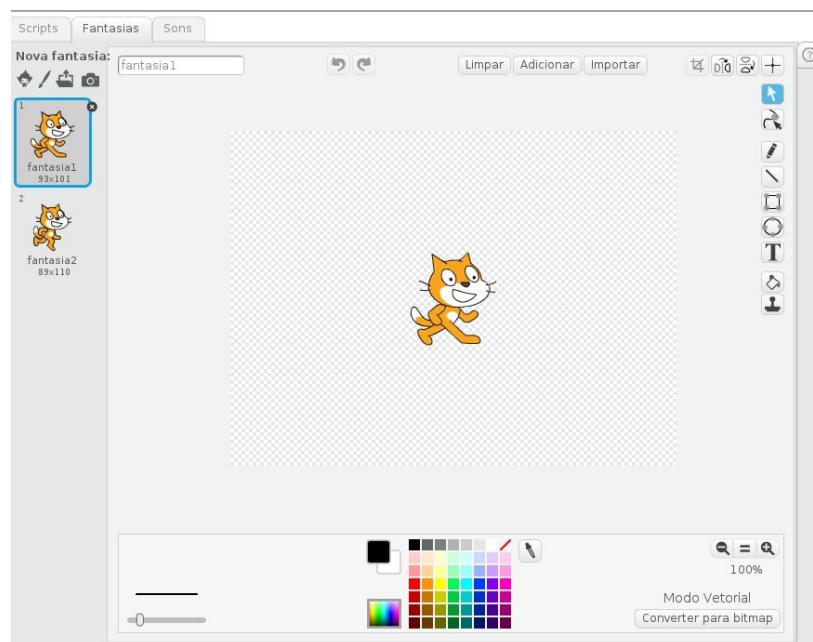
- **Mudança de Tipo de Edição**



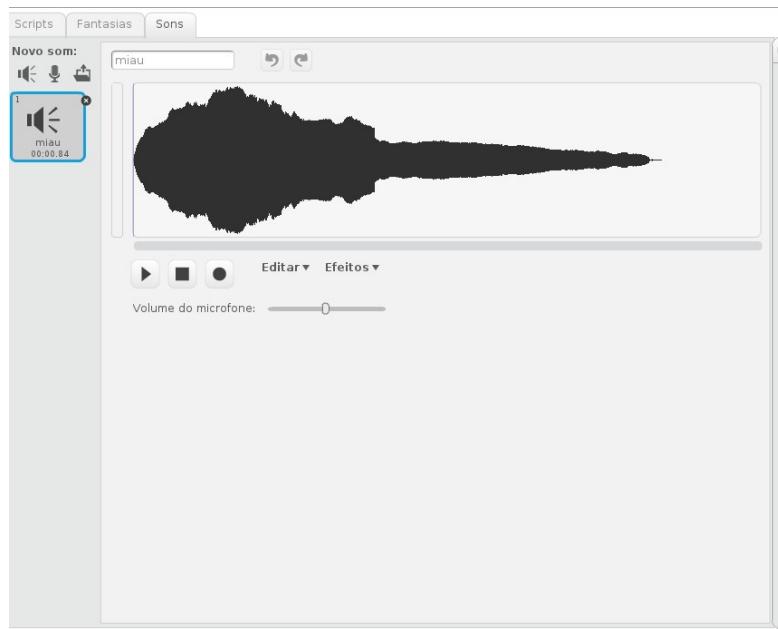
→ *Scripts*: aba de edição de *scripts*;



→ *Fantasias*: aba para edição das fantasias do objeto;



→ *Sons*: aba para criação ou edição de sons;



- **Categorias de Comando**



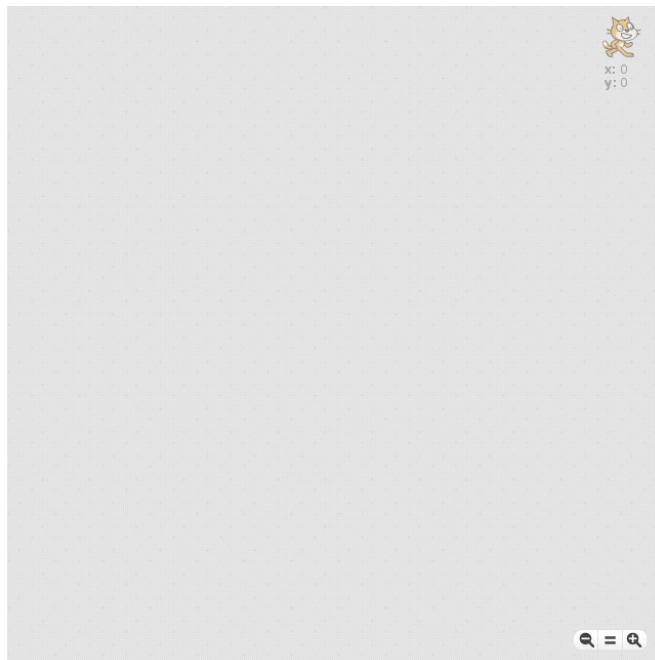
Mostra todos os tipos de blocos utilizáveis na ferramenta.

- **Blocos de Comando**



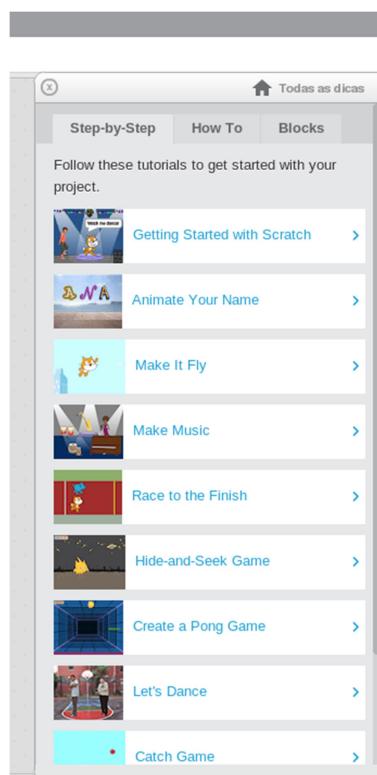
Onde aparecem os blocos da categoria selecionada.

- **Edição de Scripts**



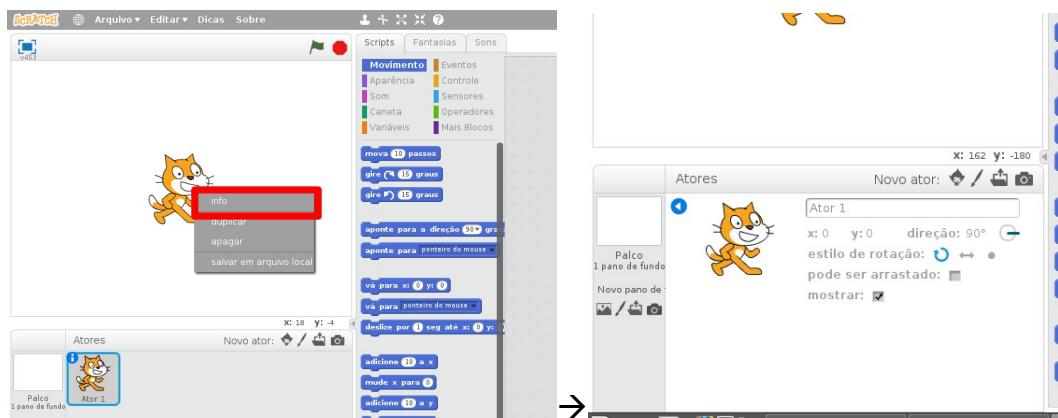
É onde se faz as montagens dos algoritmos com os blocos. Essa área pode variar entre aba de edição de scripts, de fantasias ou de sons (como mostrado acima).

- **Ajuda**



Aba que aparece quando se clica no canto direito da janela da ferramenta ou no botão ajuda (nos botões de edição de objeto). Essa aba tira dúvidas sobre os blocos, possui tutorial passo a passo e uma aba chamada “Howto”, que ensina como fazer coisas específicas no projeto de animação, jogo, música, etc.

Obs.: Diferentemente da versão antiga, onde as configurações do *sprite* se encontram à vista quando clicado nele, na nova versão tem-se que clicar com o botão direito no personagem e logo após em “info”, como mostrado nas imagens:



4. Movimento

O movimento é um recurso indispensável em um jogo ou em uma animação. Na paleta “Movimento” nas categorias de blocos do *Scratch* contém todos os tipos de blocos de movimento que se pode usar em um projeto. Eles são:



Contanto, falar-se-á somente dos principais por meio de exemplos.

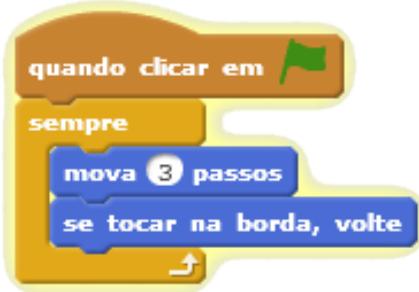
Exemplo 4.1

Peixes no fundo do mar.

Abra o *Scratch* e exclua o gato do palco. Logo após, adicione três peixes do próprio repositório do programa, se desejar, diminua o tamanho deles usando o *reduzir* na área de edição dos objetos. Coloque também um fundo de oceano (este pode ser encontrado no próprio programa, também). Na info de cada peixe, configure-o para ter o estilo de rotação somente para os lados e defina uma direção qualquer. Deve ficar parecido com isso:



Agora, adicione o seguinte *script* a eles:



Os blocos azuis são achados na paleta ‘movimento’, oblocolaranja escuro na paleta ‘eventos’ e o claro na ‘controle’. Quando acabar, aperte na bandeira verde. Animação finalizada!

Explicando os blocos de movimento usados no exemplo 4.1

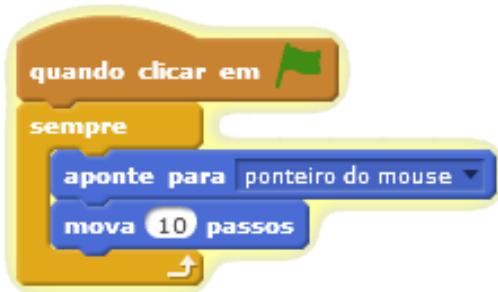
“Mova () passos”: move o *sprite* na direção pré-determinada na quantidade de passos definida pelo usuário.

“Se tocar na borda, volte”: bloco usado para impedir que o *sprite* fique preso na extremidade do palco.

Exemplo 4.2

Protegendo a cidade.

Procure na internet uma imagem de uma cidade qualquer, mas, com vista de cima. Entre no *Scratch* e coloque essa imagem como imagem de fundo do palco. Exclua o gato e adicione o “*Cat1 Flying*” de dentro da biblioteca do programa. Inclua o *script* no gato:

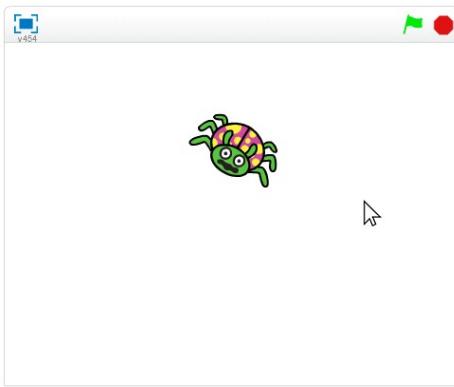


Aperte na bandeira verde e divirta-se.

Explicando o bloco de movimento usado no exemplo 4.2

“Aponte para ponteiro do mouse”: sempre que o programa executa esse bloco, o *sprite* ao qual ele foi adicionado aponta para o *mouse*. Na base que se adiciona objetos na cena, eles aparecem na caixa de seleção do bloco para ser usado como ponto de referência.

IMPORTANTE: Um objeto tem que ser desenhado de frente para a direita para parecer estar olhando para o objeto que o usuário o manda olhar. Exemplo: se o *script* do exemplo 2 for adicionado ao *sprite* ‘Ladybug2’ –encontrado no repositório- ele seguirá o *mouse* de lado e não olhando diretamente para ele. Como mostrado na imagem:

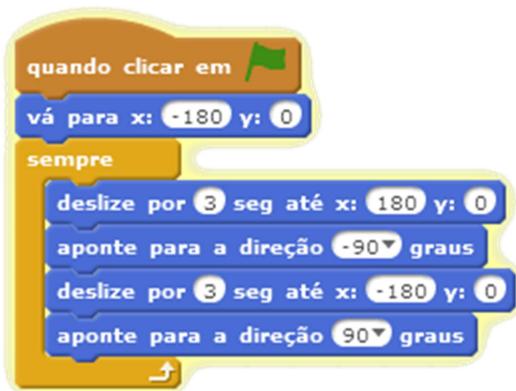


Isto é causado por o comando “Aponte para...” ser relativo aos 90º do objeto e não ao que está representado na imagem.

Exemplo 4.3

Dando vida a um fantasma.

Abra o *Scratch* e remova o gato inicial. Da biblioteca do programa, insira o ator “*Ghost2*” da aba “Imaginários”. Nas informações do fantasma, mude o estilo de giro para “esquerda-direita”. Inclua o seguinte código no *sprite*:



Logo após apertar a bandeira, observe como o fantasma desliza pelo palco.

Explicando os blocos de movimento usados no exemplo 4.3

“Vá para x: () y: ()”: esse comando faz os valores de x e y (do plano cartesiano) assumirem para valores da escolha do usuário, ou seja, se quando quiser que o personagem vá para um lugar específico do palco usa-se esse comando. Uma alternativa para ele são os comandos “Mude x para ()” e “Mude y para ()”, com a diferença desses mudarem os valores separadamente.

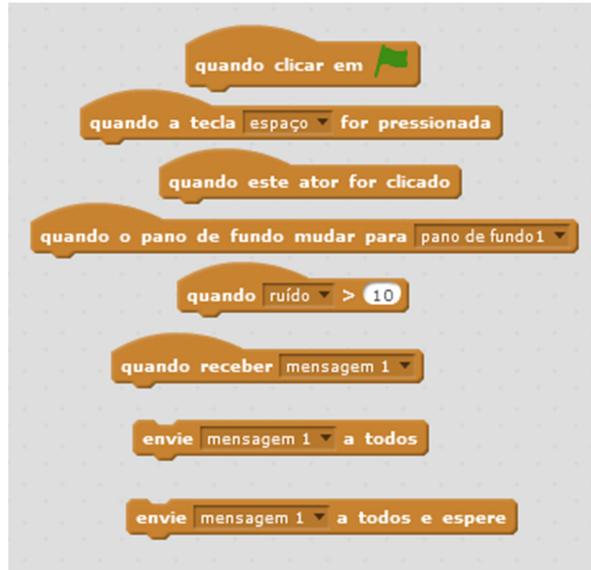
“Deslize por () seg até x: () y: ()”: faz uma animação de deslizamento no personagem até o ponto do plano que o usuário escolheu. O tempo configura a velocidade que ele vai percorrer esse caminho.

Outros comandos relevantes

- “Adicione () a x” e “Adicione () a y”: sempre que esses dois comandos forem executados, será adicionado uma quantidade pré-definida aos valores de x e y, respectivamente. Ou seja, se toda vez que o personagem bater em outro, por exemplo, ele adiciona 3 passos ao x do primeiro personagem, independente de onde esse personagem esteja no palco. Pode-se, também, colocar um valor negativo no bloco, o que implicaria de ele “remover” no lugar de “adicionar” um valor ao x ou ao y.

5. Eventos

Os blocos de eventos são usados para iniciar um pedaço de código (ou um código inteiro) a partir de um evento. Esses eventos podem ser uma tecla ou um ator clicados, uma fala, etc. São eles:



Os dois eventos mais usados são os dois primeiros: “Quando clicar em *Início*” e “Quando a tecla *espaço* for pressionada”.



: o código abaixo desse bloco começará imediatamente após apertar a bandeira verde acima do palco.



: esse bloco pode ter vários parâmetros. Eles são: setas (cima, baixo, esquerda e direita), letras (a, b, c, d,..., y e z), números (1, 2, 3,..., 9 e 0), espaço e qualquer tecla. Pode ser usado, por exemplo, para andar para a direita: “quando a tecla *seta para a direita* for pressionada’, ‘aponte para a direção 90 graus’, ‘ande 10 passos’”.

6. Controle



Os blocos de controle são aqueles que, literalmente, controlam todo o código e toda a execução dele. Nessa categoria encontram-se repetições, paradas, condicionais, etc.

Exemplo 6.1

Voando em círculos.

Após abrir o programa e deletar o gato, escolha o “*Bat1*” da biblioteca do *Scratch*. Inclua, então, este código ao morcego:



Percebe-se, ao executar-se o código, que o morcego gira uma volta completa e, ao terminar, uma volta menor.

Explicando os blocos de controle usados no exemplo 6.1

“Sempre”: tudo que se encontra dentro desse bloco ficará se repetindo para sempre – ou até um “pare” ser executado -.

“Se () então”: o pedaço de código dentro desse bloco somente será executado se a afirmação declarada for verdadeira. Nesse caso, a afirmação é “direção = 90”, ou seja, o “repita 24 vezes” é executado somente quando a direção do morcego é 90°.

“Repita () vezes”: tem a função igual ao *sempre*, mas com números contados de vezes, e não para sempre.

Outros comandos relevantes

“Se () então... Senão”: esse bloco mantém a mesma lógica do “Se () então”, mas com a adição de um “senão”, ou seja, se algo for verdadeiro, ele o que está dentro da parte do “se”, mas, se não for verdadeiro, ele executa o que está dentro do “senão”.

“Repita até que ()”: imita a execução do “sempre”, mas para de executar quando a afirmação se torna verdadeira. Por exemplo, um personagem anda pelo palco até chegar a uma porta.

“Espere () seg”: deixa um período x de tempo em que o *sprite* não faz nada.

7. Aparência



Os blocos de aparência são importantes para a troca de traje e para a fala dos personagens na cena. Eles também trocam a cor e colocam efeitos nos *sprites*, tais como: *olho de peixe*, *mosaico*, *fantasma*, etc.

Exemplo 7.1

Dando vida a um fantasma 2.

Usando uma cópia do projeto “*Dando vida a um fantasma*”, abra o *scratch* e mude o código do fantasma desse modo:





O fantasma agora ganhou a animação de aparecer e desaparecer, além de mudar a forma dele a cada vez que ele faz isso!

Explicando os blocos de aparência do exemplo 7.1

“Esconda”: faz o personagem desaparecer do palco.

“Próxima fantasia”: troca o traje do personagem em ordem (se houver outro), ou seja, se houver três fantasias, o bloco troca do primeiro para o segundo e, se for executado novamente, do segundo para o terceiro e assim por diante.

“Mostre”: faz o sprite aparecer no palco (caso ele esteja escondido).

Exemplo 7.2

Conversa entre amigos.

Abra o Scratch e adicione dois personagens (se quiser pode usar o gato como um deles). Coloque-os em uma distância curta um do outro. Defina então como plano de fundo a imagem chamada “castle3” da própria pasta do Scratch. Após isso, adicione o seguinte código nos personagens:



Personagem 1

Personagem 2

Ao clicar na bandeira verde, você verá os dois conversando.

8. Operadores



Os blocos de operadores são usados para comparações, contas matemáticas, geração de números aleatórios, etc. Eles são usados encaixados em outros blocos (como visto no exemplo 7.1, por exemplo). Eles podem ser encaixados dentro deles mesmos também, como mostra a figura a seguir:



Obs.: cada bloco equivale a um parêntese, ou seja, ele resolverá, nesse caso, primeiro o bloco “() - ()” e o bloco “() / ()” para depois resolver o “() * ()”. Essa expressão teria essa forma:

$$[(10 - 5) * (6 / 2)] + [(6 - 3) * (10 / 5)] = 21$$

Os blocos de comparação “() < ()”, “() = ()” e “() > ()” são usados para saber se uma coisa é verdade ou não para, por exemplo, ser usado dentro de um “se () então” ou dentro de um “repita até que ()”. O símbolo “<” quer dizer ‘menor que’, o “=” quer dizer ‘igual a’ e o “>” quer dizer ‘maior que’.

Exemplo 8.1

“7 < 10”: ‘sete é menor que 10’. **Verdadeiro**

“4 = 5”: ‘quatro é igual a 5’. **Falso**

Já os blocos de comparação lógica “() e ()”, “() ou ()” e “não ()” seguem uma regra diferente. Estas são mostradas nas tabelas a seguir:

E		OU		NÃO	
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	F	V	V
F	F	F	F	F	F

Entradas
Saída

Ou seja, o bloco “() e ()” somente retornará verdadeiro se as duas entradas dele forem verdadeiras.

Exemplo 8.2



O bloco “() ou ()” retorna verdadeiro se qualquer uma ou as duas entradas forem verdadeiras.

Exemplo 8.3



O bloco “não ()” inverte o valor de saída, ou seja, se a entrada é positiva, ele a torna negativa. E vice-versa.

Exemplo 8.4



Outros blocos relevantes

“Resto de () por ()”: ele faz a divisão do primeiro número pelo segundo e diz o resto da divisão. Isso ajuda na hora de descobrir se um número é par, por exemplo, pois qualquer número par dividido por 2 tem o resto 0.

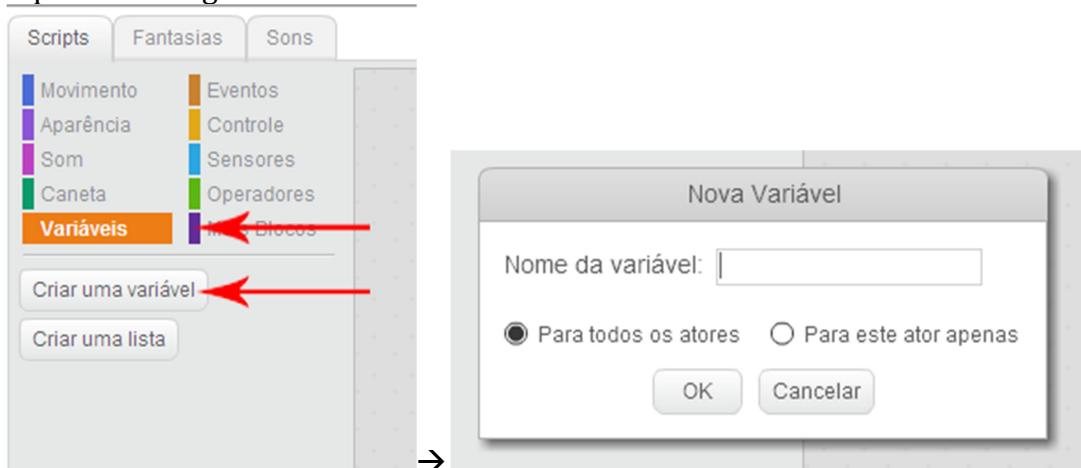
“Arredondamento de ()”: tem como entrada um número decimal (com vírgula) e como saída o número inteiro mais próximo dele. Um exemplo disso é o número 9,3, ele o arredondaria para 9. Já o número 9,7, para 10. **Obs.:** um número terminado em “,5” é arredondado para cima. 9,5 = 10.

“Raíz quadrada de ()”: esse bloco contém funções matemáticas, tais como: logaritmo, logaritmo natural (ln), seno, cosseno, tangente, etc.

9. Variáveis

Variável é uma palavra que contém qualquer valor dentro dela. Por exemplo: a contagem de pontos (o *score*) de um jogo, quanto mais se joga, mais pontos têm-se dentro dessa variável. Outro exemplo bem comum é a variável *x* da matemática: " $x + 5 = 20$ " nesse caso, *x* valeria 15, mas pode haver casos em que vale 55 ($100 - 45 = x$), por exemplo. Ou seja, uma variável é uma palavra que guarda um valor e, esse valor, pode *variar*.

Para criar uma variável no *Scratch* basta ir à paleta "variáveis" e clicar em "criar uma variável". Após clicar nisso, vai aparecer uma janela. Nela, coloca-se o nome da variável e seleciona-se se é uma variável de todos os atores ou somente do que está selecionado. Clique em "OK" quando configurar tudo.



Irão aparecer, então, alguns blocos referentes à variável recém criada.



O primeiro bloco ("Variável") contém uma caixa de seleção do lado. Essa caixa é para selecionar se o valor da variável aparece ou não no palco. Esse bloco pode ser usado dentro de outras funções.

Exemplo 9.1



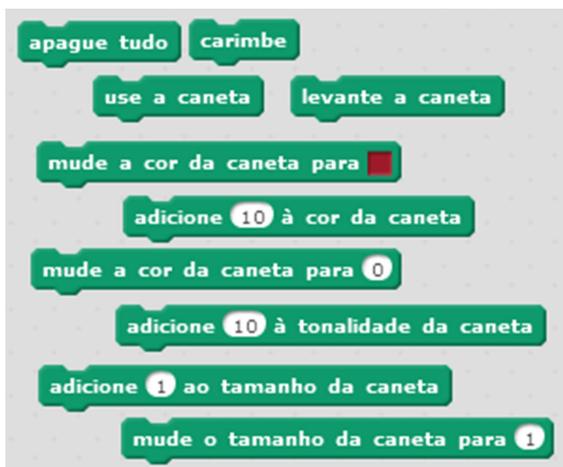
O bloco “mude *Variável* para ()” é para mudar o valor da variável. Por exemplo, no início de um jogo, ninguém começa o jogo com pontos, logo, põe-se esse bloco para zerar os pontos.

“Adicione a *Variável*()” serve para somar o valor de entrada à variável. O valor de entrada pode, também, ser negativo, logo, o valor seria subtraído da variável.

“Mostre variável *Variável*” e “Esconder variável *Variável*” são para quando a caixa de seleção da variável estiver desmarcada e precisar mostrar e esconder o valor da variável.

Obs.: uma lista é um conjunto de variáveis, ou seja, são várias variáveis com o mesmo nome.

10. Caneta



A caneta serve para marcar o caminho por onde seu personagem passa ou para carimbar ele no plano de fundo. A seguir explica-se sobre os blocos mais importantes dessa paleta:

“Apague tudo”: todos os riscos (ou carimbos) existentes no *background* são apagados.

“Carimbe”: imprime a imagem do *sprite* na imagem de fundo do palco.

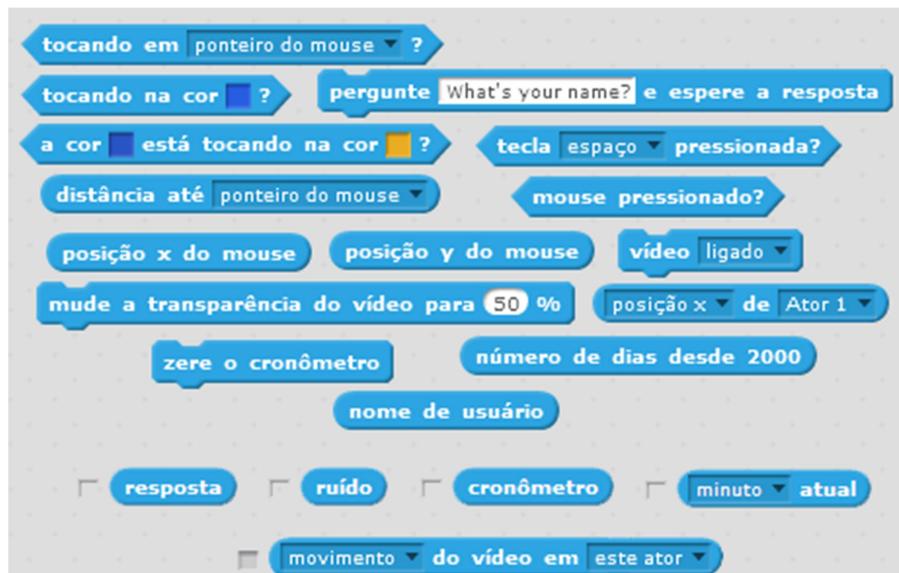
“Use a caneta”: serve para “encostar” a caneta no plano de fundo, como se tivesse, por exemplo, encostando a caneta no papel para escrever.

“Levante a caneta”: bloco com a função contrária ao bloco anterior, ou seja, serve para “desencostar” a caneta do plano de fundo.

“Mude a cor da caneta para ()”: define a cor que a caneta irá riscar no *background*.

“Mude o tamanho da caneta para ()”: define a grossura do risco da caneta.

11. Sensores



Os sensores são o que auxiliam em todo processo de programação do jogo. Com eles é que se faz a leitura de algo que o jogador escreveu, consegue-se a distância de duas coisas no palco, verifica-se se algo está sendo clicado, etc.

Exemplo 11.1

Dando nome ao gato.

Abra o *Scratch* e adicione o seguinte *script* ao gato:



Note que abre uma aba no palco para você digitar o nome do *sprite*.

Explicando os blocos de sensores usados no exemplo 11.1

“Pergunte [] e espere a resposta”: abre a aba de leitura para o jogador digitar o que está sendo pedido.

“Resposta”: guarda a última coisa digitada pelo jogador. Este bloco pode ser ativado para aparecer o conteúdo no canto do palco.

Obs.: Perceba que no código usado no exemplo 11.1, se outra resposta for lida do jogador, o nome do gato se perderá. Isso pode ser evitado se guardar o nome em uma variável. A adaptação do código ficará assim:



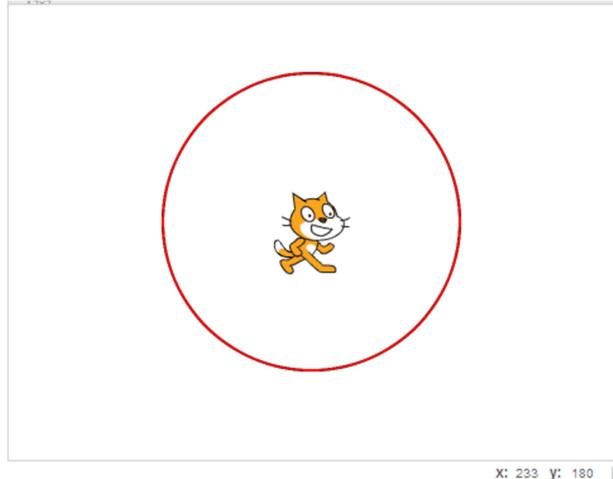
IMPORTANTE: no final do primeiro argumento do bloco “junte [] com []” deve ser colocado um espaço, pois, caso contrário, ele juntará os argumentos em uma única palavra.



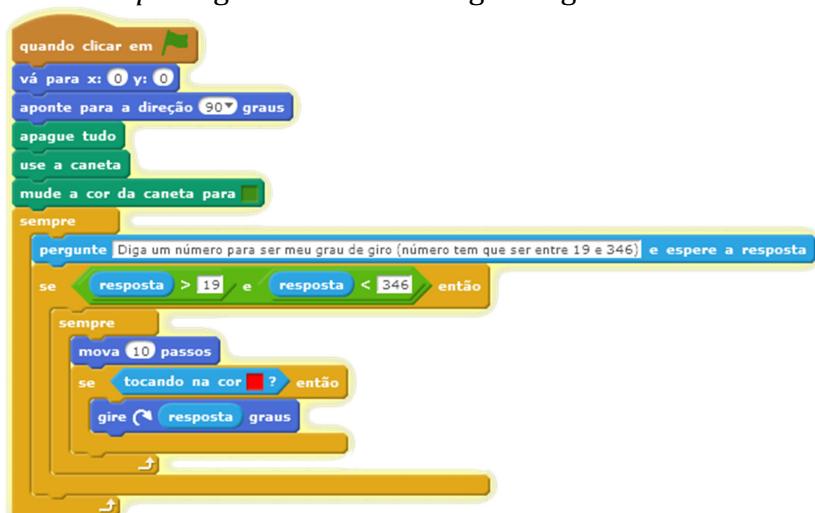
Exemplo 11.2

Desenhando uma forma qualquer.

Abra o programa e diminua o tamanho do gato. Edite o plano de fundo do palco de modo que fique um círculo vermelho desenhado no centro.



Logo após, abra o *script* do gato e insira o código a seguir:



Sugestão: teste com 190, 210 e 290.

Perceba que ele limita o número lido entre 19 e 346. Isso se deve à velocidade que ele passa pela linha vermelha se o grau for menor ou igual a 19 e maior ou igual a 346, não dá tempo de girar o suficiente para o *sprite* ficar dentro do círculo. Se quiser, diminua a quantidade de passos e tire o “se () então” presente após a pergunta e tire também o “sempre” colocado antes da pergunta.

Explicando o bloco de sensor usado no exemplo 11.2

“Tocando na cor []?”: quando qualquer parte do personagem encosta em qualquer coisa da cor selecionada, ele da sinal de positivo e executa o bloco da condição. Um bloco com o mesmo propósito, mas não considerando uma cor, é o “tocando em ponteiro do mouse?”. Este vem por padrão o “ponteiro do mouse” e “borda” na caixa de seleção, mas, à medida que vai se adicionando coisas ao programa, ele vai atualizando a lista.

Outros blocos de sensores relevantes

“Mouse pressionado?” e “tecla espaço pressionada?”: servem para a verificação de teclas ou cliques do mouse. Podem ser introduzidas em “se () então”, “repita até que ()”, etc.

“Posição x do mouse” e “posição y do mouse”: retornam o valor da coordenada cartesiana x e y, respectivamente. Podem ser usadas para fazer algum objeto da cena seguir o mouse.

12. Exercícios

- 1) Faça uma animação que simule uma sinaleira. Use blocos de comando (como “espere () seg”) para simular o tempo de cada luz. Você deve desenhar o *sprite* e cada luz acesa deve ser uma fantasia desse mesmo personagem.
- 2) Anime uma conversa entre uma bruxa e um feiticeiro. A história deve ter, no mínimo, 5 falas para cada personagem.
- 3) Produza uma animação que o *sprite* ande de acordo com a seta do teclado pressionada. Disponibilize também o uso da caneta e faça a tecla “d” descer a caneta e a tecla “s” subir a caneta.
- 4) Simule a cena em que o Darth Vader conta ao Luke Skywalker que ele é o pai dele. Eles devem dizer as famosas frases “Luke, eu sou seu pai” e “NÃÃÃÃOOO!!!” para, então, o Luke se jogar no abismo.
- 5) Elabore um programa que leia um número do teclado, armazene-o em uma variável e faça o gato dizer a tabuada referente a esse número.