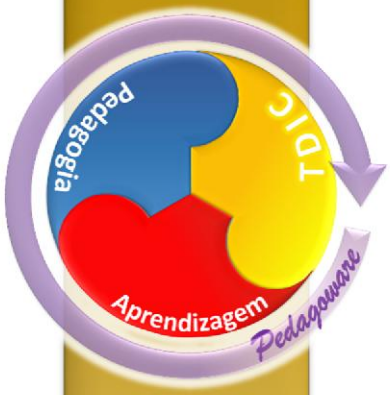
Digital

Pedagoware: TIC (SCRATCH) + Pedagogia + Aprendizagem



COMPETÊNCIAS DA TECNOLOGIA DIGITAL DA INFORMAÇÃO E COMUNICAÇÃO NA EDUCAÇÃO,

mediadas pela interface de programação Scratch

Marco Antonio G T da Silva – Mestrando PPGCN
marcoagts@gmail.com

Sérgio Luis Cardoso – Prof. Dr.

As novas tecnologias proporcionam convergência entre várias áreas. A área de educação sempre buscou integrar as Tecnologias da Informação e Comunicação (TIC) às suas atividades para aplicar ao ensino. A TIC no formato digital, ou seja, a Tecnologia Digital da Informação e Comunicação (TDIC), torna-se cada dia mais expoente socialmente. Uma das ferramentas importantes da TDIC é o ambiente de programação, pois permite produzir infinitos recursos. Entretanto, as ferramentas de programação sempre foram afeitas aos "escovadores de bits" e poucos docentes dominam esses recursos. Nesse contexto, o docente vive essa dicotomia: a da necessidade de usar recursos personalizados em sala e, não reunir conhecimentos para se desenvolver, por desconhecer sintaxe de programação. Incorporar ferramentas que permitam desenvolver uma interação com conceitos de forma visual, facilita a atividade do docente. Possibilitar ao docente o desenvolvimento e, se for o caso, o professor poderá ensinar o conhecimento ao aluno, para que ele utilize a programação, e concretize os conceitos pesquisados em ambientes digitais, tem sido fator de pesquisas e publicações que geralmente apontam alto grau de sucesso. O problema é ensinar ao docente uma ferramenta simples e de fácil acesso. O atual patamar da TDIC nos possibilita resolver essa "desconexão". Por isso, esse material é planejado para apresentar uma interface de programação e fundamentos que irão possibilitar a tarefa para trabalhar com os atuais nativos digitais. Estamos também propondo transgredir layout e os limites físicos das salas de aulas convencionais, pois sabemos que o docente necessita de uma formação com a Educação "sem" a Distância. Assim, todo o material foi pensado para uma interação por intermédio do Ambiente Virtual de Ensino e Aprendizagem. Essa é a nossa proposta: criar recursos para iniciar a educação 2.0 e caminhar para a docência do século XXI. E, faremos isso explorando os recursos do Scratch 2.0 como uma ferramenta para o planejamento pedagógico.

Sumário

Conheça todas as teorias, domine todas as técnicas e tecnologias ...
Mas ao tocar uma alma humana, seja apenas outra alma humana.
Carl Gustav Jung

Apresentação

Este material foi planejado para apresentar o Scratch a quem não tem conhecimento de programação. O uso da ferramenta de programação se propõe a torná-la um coadjuvante na sala de aula, rompendo com os limites físicos e utilizando-se do imaginário e da criatividade, conforme é permitido com a Tecnologia Digital da Informação e Comunicação (TDIC), pois acreditamos que a educação do século XXI vai além dos conceitos transcritos no papel e no quadro, pelo fato de os participantes desse novo momento estarem cada vez mais preparados para serem colaborativos na sua própria formação, dependendo apenas de nós, para incentivá-los e direcioná-los.

Organização

Nosso curso é desenvolvido em tópicos. Como o objetivo do curso é conhecer o ambiente de programação Scratch, iremos discutir essa ferramenta que possibilita, além de desenvolver simuladores virtuais, criar processos de pesquisa como forma de aprendizagem.

Cronograma das atividades participativas

Semana	Conteúdo	Atividade	Finalidade
1ª	Apresentação do Scratch	Apresentação da interface do Scratch	Conhecer a interface de trabalho (Palco, spriter e blocos)
2ª	Ambientação Scratch	Atividade com Scratch (fórum)	Ambientação com a interface Scratch (eventos, fantasias e áudio)
3ª	Implementação de virtualidade aumentada	Atividade com Scratch (fórum)	Criar uma interação imersão virtual (webcam, mensagens entre eventos, personagens e bloco variável).
4ª	Uso da Caneta e interação com o usuário	Atividade com Scratch (fórum)	Manipular a “caneta”, duplicar spriter e controle de variáveis.
5ª	Menu e pano de fundo	Atividade com Scratch (fórum)	Desenvolver um menu inicial, mudar o pano de fundo, duplicar spriter e controlar efeitos.
6ª	Reusabilidade de arquivos e evento clone	Atividade com Scratch (fórum)	Reutilizar e adaptar projetos prontos, criar e controlar o clone.
7ª	Vinhetas de projetos e blocos personalizados	Atividade com Scratch (fórum)	Criar efeitos de transição, blocos personalizados e efeito para mouse.
8ª	Material potencialmente significativo	Pensando o projeto	Planejar um projeto de TDIC dentro de uma teoria educacional
9ª	Portabilidade com Scratch	Publicação do projeto	Publicar o projeto e incorporar em outros ambientes virtuais.

Por que desenvolver um simulador?

O recurso tecnológico amplia o potencial humano, contudo a "educação tradicional" observa a tecnologia de forma passiva (SOFFNER, 2013). O emprego de tecnologia pode alterar o quadro da educação, aplicando recursos como simulações para representar os conceitos bibliográficos Soffner (2013). O autor ressalta ainda que não é a proposta piagetiana, com o uso de computador em sala, mas sim uma questão de trabalho sistemático de modelagem e simulação. É importante comentar que a simulação virtual possui alto potencial de acesso, com qualidade e baixo custo financeiro (BAKIA, 2000; MA; NICKERSON, 2006).

A simulação virtual permite converter mecanismos físicos, abstratos, em mídias, representando esquemas ou comportamento que permitam o entendimento do conceito da literatura. Dessa forma, possibilita produzir um sistema hipotético, deduzido do mundo real, contudo, consciencioso entre a lógica e a concepção acadêmica (DASGUPTA; RESNICK, 2014; MA; NICKERSON, 2006; DORNELES, 2010). Desse modo, a simulação realiza operações e sistemas complexos, por intermédio de processos pictóricos, que são visualmente simples e viáveis economicamente para diversas situações.

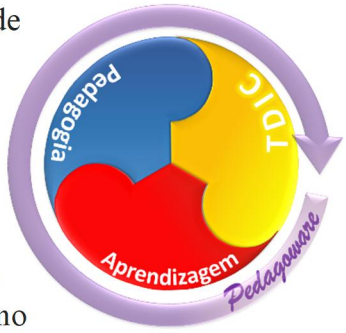
Valente (2014) relata que a aplicação das TDIC na educação, por intermédio da programação ou da simulação de fenômenos, realiza tarefas onde o aprendiz descreve ideias na forma de instruções, usando os recursos de comunicação específicos para cada uma dessas tarefas. Para Valente (2014), o uso de programação em sala é um recurso importante, que ajuda na formulação de conteúdos e lógicas.

O ambiente de desenvolvimento do Scratch permite criar a simulação por interposição de várias mídias (RESNICK et al., 2009). O projeto do ambiente Scratch, além de criar simulações, pode ser considerado como "mini *web-app*" (pequenos aplicativos para *web*) (DASGUPTA; RESNICK, 2014), fator que irá auxiliar na portabilidade de um projeto.

Por todas essas razões e pela facilidade que proporciona a interface do Scratch, é que resolvemos pela adoção da linguagem para criar a competência de programação e o desenvolvimento simulador digital.

Figura 1 – Pedagoware

O **símbolo**



O símbolo foi pensado e criado levando-se em conta as ideias de tecnologia, pedagogia e tecnologia, ligadas ao conteúdo principal: o Scratch. E o resultado é a criação de uma base em que as imagens se encaixam.

"Dissecando" a imagem central, é possível observar uma forma circular, em que não há uma informação primordial, nem inicial, mas um assunto diferenciado. Nesse formato, ocorre o encaixe entre os contextos da Tecnologia Digital da Informação e da Comunicação (TDIC), da Pedagogia e da Aprendizagem. Destacamos que o termo ensino não aparece.

As formas dos blocos são de encaixe, lembrando o Scratch, coloridas nas cores primárias, lembrando que **será vista apenas a base, mas cada informação tem uma engrenagem**.

Observando o termo *Pedagoware*, ele está circulando e passeia por todos os conceitos. A cor do termo pedagoware está ligada à da formação do pedagogo.

A mensagem é que o uso da tecnologia seja planejado para aula, dentro de uma pedagogia para propiciar a aprendizagem.

Esse é o objetivo do curso: apropriar-se da tecnologia de programação propiciada pelo Scratch, para romper com a educação do século XIX e formalizar o docente 2.X e assim retirar de cena o professor protagonista de papel e lápis.

Ressalta-se que as tecnologias devem ser compreendidas com ideal educacional e "as TDICs podem ser extremamente úteis como ferramentas cognitivas, desempenhando diferentes papéis" (VALENTE, 2014).

A proposta

O uso do Scratch deve ser visto com a postura do mediador da construção do conhecimento no aluno (VALENTE, 2014).

Não se pretende de forma alguma acreditar que criar a competência para desenvolver simuladores digitais, perfeitos ou não, vá fazer com o aprendiz formule o conhecimento do "zero" sem a presença do mediador. Porém, como afirma Valente (2014) "a ação educacional consiste justamente em auxiliar o aprendiz, de modo que a construção de conhecimento possa acontecer. Isso implica criar ambientes de aprendizagem onde haja tantos aspectos da transmissão de informação quanto de construção".

Assim sendo, também mencionaremos sobre a forma de planejar o desenvolvimento do recurso da tecnologia digital, contudo ressaltamos que o foco principal é desenvolver competência para o uso da tecnologia digital de programação em blocos, usando o Scratch.

A apostila

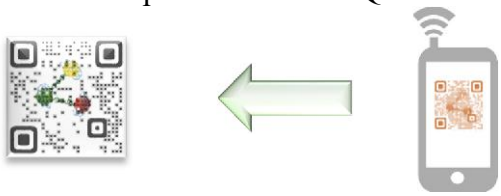
É importante ressaltar que **esse material não substitui de forma alguma a interação**, apenas tem por objetivo tornar mais prático o nosso curso.

Esse material tem como proposta o estudo de recurso digital e na plataforma serão apresentados recursos *on-line*, por isso foram incluídos aqui *tags,* que servem para "*linkar*" o usuário do papel (esse material) com a *internet*. A *tag,* quando for lida por um dispositivo móvel (*smartphone* ou *tablet*), é capaz de abrir *links* da *internet*. Assim, estamos inserindo na leitura do material estruturado de forma física, a interatividade existente no ambiente da *web*.

Essas *tags* possuem o nome de QRCode e para serem executadas necessitam de leitor (programa específico) disponível para ser baixado em cada aparelho móvel e também de acesso à internet, pelo aparelho que realizar a leitura. Desse modo, quando a imagem for identificada, possibilitará fazer uma leitura com seu celular ou *tablet* do *link* na *internet* e ver o vídeo ou acessar o texto.

Mas não se esqueça de que é necessário o programa para leitura do QRCode, instalado no seu dispositivo móvel, acesso à *internet* e que a impressão utilizada seja de boa qualidade. Ao acionar o aplicativo (Leitor de QRCode) no dispositivo móvel e apontá-lo para a imagem, o dispositivo vai diretamente para o site da informação.

Figura 2 – Exemplo de leitura de QRCode



QRCode – Modelo **Leitor (dispositivo móvel)**

Ambiente Virtual de Ensino e Aprendizagem

Escolhemos o Ambiente Virtual de Ensino e Aprendizagem (AVEA), pois um curso com proposta de Ensino a Distância (EaD), ou "Ensino sem Distâncias Geográficas", ou ainda Ensino Virtual, caracteriza-se por romper a relação presencial entre o mediador e o participante, entre o espaço físico geográfico e a temporalidade. Para Schlosser (2007), essa ruptura faz com que o participante da EaD decida sobre seu processo de formação, de forma autônoma e independente, aliado às facilidades de interação via Internet, viabilizando uma formação que é muito difícil de ser realizada presencialmente (VALENTE, 2014).

Vale ratificar que a metodologia EaD agrega valores tecnológicos que tendem também a se tornar mais um marco na evolução e reformulação na estrutura dos cursos presenciais. Também, que a "EaD pode utilizar abordagens pedagógicas, que exploram os verdadeiros potenciais que as TDICs oferecem, ao facilitar não somente o aprofundamento da interação professor-aprendiz, mas também entre aprendizes" (VALENTE, 2014).

Assim, ao utilizar os recursos da *web* 2.0, estamos democratizando o processo de ensino e aprendizado com uma variedade de ferramentas e serviços disponíveis de forma gratuita na rede. O formato da *web* 2.0, com os vários recursos de interação, suas linguagens e *plug-ins,* cria uma interação em tempo real, porém demanda uso de banda larga, que já está disponível em dispositivos móveis.

Ainda considerando os recursos apresentados, a utilização do ambiente Moodle possibilita a submissão, envio, compartilhamento, organização de material e tarefas para alunos geograficamente distribuídos, e também promove *feedbacks* durante o curso, além de permitir a organização do tempo de estudo por parte do participante aluno.

A finalidade dessa etapa é apresentar todos os recursos da interface Scratch, que iremos trabalhar nesse curso. Dividimos o ambiente de programação em três áreas: Palco, Blocos e Sprites.

Apresentação do Scratch

O Scratch é um ambiente gráfico de programação acessível a todos. Foi publicado em 2006, pelo grupo de pesquisa Lifelong Kindergarten, do laboratório de Mídia do Instituto de Tecnologia de Massachusetts (*Massachusetts Institute of Technology* - MIT), (http://llk.media.mit.edu), com o apoio financeiro da *National Science Foundation*, Microsoft, Intel Foundation, Nokia, e o consórcio de investigação do MIT (FORD JR, 2009). O projeto inicial é baseado no *software* Logo, desenvolvido na década de 60, por Seymour Papert. Permite desenvolver raciocínio, lógica, imaginação, autonomia, investigação e criação.

O Scratch é uma interface acessível para todas as idades, que proporciona uma programação em blocos (linhas de comandos) sob forma de encaixe, formando uma sequência lógica e exigindo pouco conhecimento de programação. O Scratch permite desenvolver histórias animadas, simulações, jogos, projetos de ciências, projetos interativos, com inserção de várias mídias (RESNICK et al., 2009; MARJI, 2014).

O recurso de programação em blocos visuais foi escolhido pela praticidade da "linguagem" e possibilidade de criar estímulos sensoriais favoráveis à aprendizagem com a inserção de imagens. As imagens utilizadas podem ser desenvolvidas com auxílio de outro *software* ou no próprio ambiente do Scratch.

Para entender o que é diferente no Scratch, vamos iniciar falando sobre a sintaxe. Para que qualquer dispositivo da área computacional funcione, existem algumas informações prévias que são inseridas no dispositivo; essas informações são denominadas programas ou algoritmos computacionais, que são as instruções. As instruções possuem uma forma de expressão específica (sintaxe) para desenvolver passos lógicos e finitos. Para entender melhor, observe o exemplo de como exibir uma mensagem na tela:

Exemplos para apresentar uma mensagem na tela:

Sintaxe no ambiente de programação	*Linguagem (ambiente de programação)*
print ('Curso de Scratch na UENF')	(Phynton)
std :: count<< "Curso de Scratch na UENF" <<*std :: endl;*	(C++)
System.out.println("Curso de Scratch na UENF");	(Java)
printf("Curso de Scratch na UENF\n");	(C)
echo "Curso de Scratch na UENF"	(Shell script)
document.write ("Curso de Scratch na UENF")	(Java script)
<php? echo "Curso de Scratch na UENF">	(PHP)

Cada interface de programação possui uma sintaxe (conjunto de regras, lógica e semântica própria). A sintaxe geralmente torna-se o maior ponto de desafio aos iniciantes na área de programação (MARJI, 2014). A dificuldade da sintaxe muda no Scratch, pois para montar um *script* é necessário usar um bloco pronto, conforme Figura 3A.

Figura 3 – Exemplo de uso do Scratch

Figura 3A - ambiente de programação

Figura 3B – mensagem na tela

Observe que no Scratch não há uma sintaxe para ser memorizada, ou seja, esse ambiente não utiliza comando, apenas os blocos. No caso dos blocos, também não há necessidade de memorização, pois são organizados e agrupados (paletas) em cores diferentes, conforme a finalidade.

O Scratch tem outras vantagens, como um repositório mantido pelo grupo de pesquisa Lifelong Kindergarten, do laboratório de mídia do MIT; isso possibilita à comunidade de programadores melhorar o desempenho da interface do Scratch e, caso queira, de seu projeto, além de permitir criar projetos *on-line* ou *off-line*.

O ambiente de desenvolvimento do Scratch não tem compilação, ou seja, após concluir o projeto, não é possível criar um arquivo executável, como um aplicativo autônomo. No ambiente Scratch, a linguagem é interpretada com uma programação dinâmica (FORD JR, 2009). Esse processo de interpretação também é um facilitador, que oferece um *feedback* imediato, que permite conferir a lógica de programação de forma rápida.

Ambientação com a interface Scratch

O 1º passo é baixar o ambiente de desenvolvimento do Scratch, no site do MIT (**https://scratch.mit.edu/scratch2download/**), destaque na Figura 4.

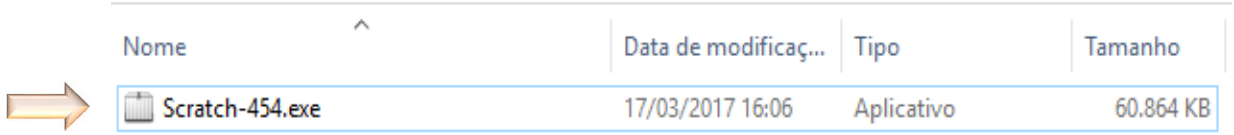
Figura 4 – *Link* para baixar o Scratch *off-line*

Link para download

No *site* do MIT também é possível encontrar os *links* para o Adobe AIR, que é necessário para executar o ambiente e alguns materiais para os primeiros passos (material no idioma inglês).

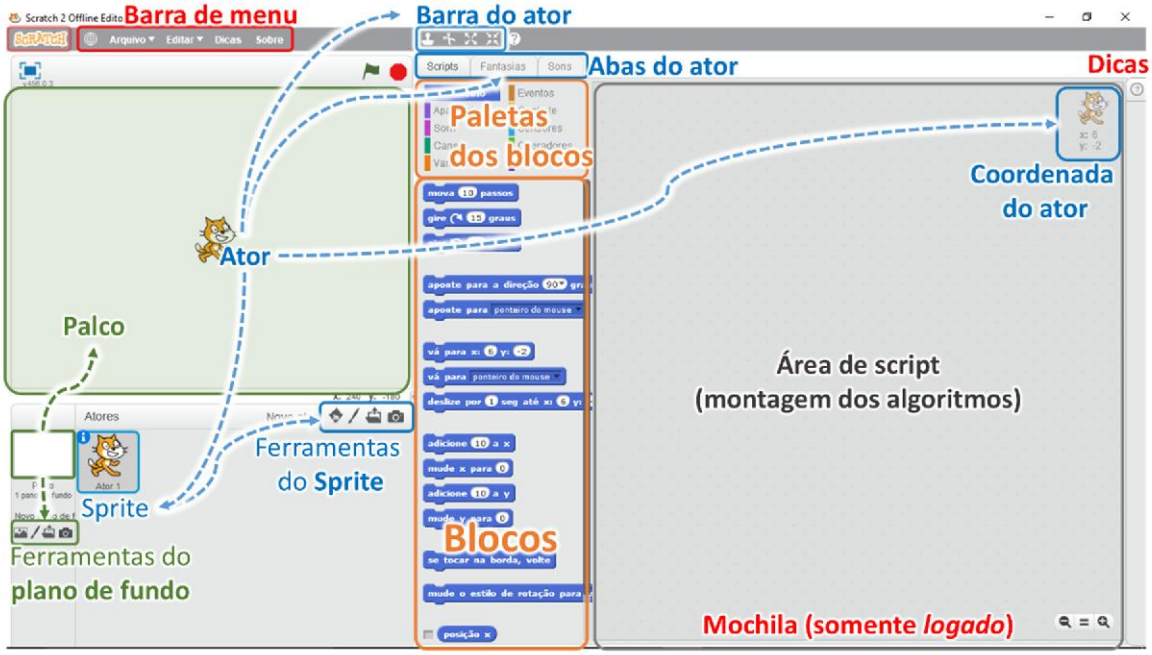
Após baixar aparecerá o arquivo "Scratch-XXX.exe" (XXX é a versão) (Figura 5). É só dar um duplo *click* para executar.

Figura 5 – Arquivo de instalação do Scratch



Depois de instalado, ao ser executado, aparecerá o ambiente de programação no Scratch (Figura 6).

Figura 6 – tela inicial do ambiente Scratch



No ambiente Scratch, existem quatro grupos importantes: (i) os *spriters* e o palco. Os *spriters* são também os atores que desenvolvem as histórias. Se houvesse uma tradução para a palavra *spriter*, poderia dizer que são as imagens que produzem os eventos gráficos de "forma mágica". No caso do Scratch são imagens bidimensionais. O palco (também denominado de *stage,* estúdio ou pano de fundo) é o ambiente onde ocorrem os eventos. O pano de fundo pode também interagir com os eventos; (ii) as abas (*scripts*, fantasias e sons), que são utilizadas para produzir o(s) evento(s) relacionado(s) ao ator, por isso, estamos denominando "abas do ator", para criar as fantasias dos *spriter*, ou seja, diferentes formas de apresentar o mesmo *spriter* e para controlar ou criar sons; (iii) os blocos que são encaixados, para produzir os *scripts* e são organizados por cores, conforme a finalidade; e, (iv) área de *script*, onde são dispostos os blocos para que os *spriters* realizem os movimentos ou efeitos. Por fim, ainda existem os comandos relativos ao funcionamento da interface, ou menu (na Figura 6 estão destacados na cor vermelha na parte superior).

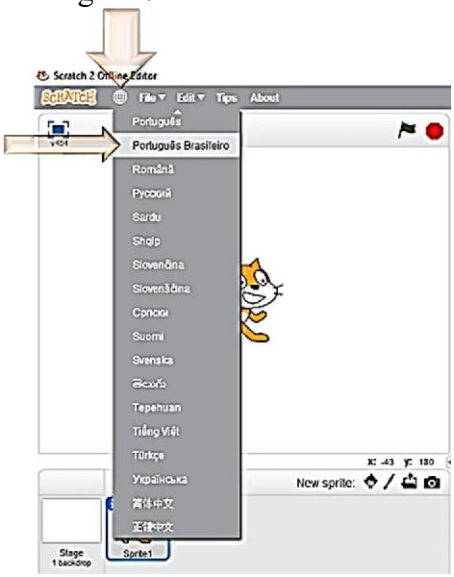
O uso dos recursos do Scratch, ou seja, a criação de um arquivo, envolve vários recursos de mídias, por isso é denominado de Projeto.

Ferramentas do ambiente

Mudança de idioma

Antes de iniciar a descrição, é possível colocar a interface em português. Para isso, basta ir na imagem do Globo, no canto esquerdo da barra de menu, e selecionar "Português Brasileiro" (Figura 7).

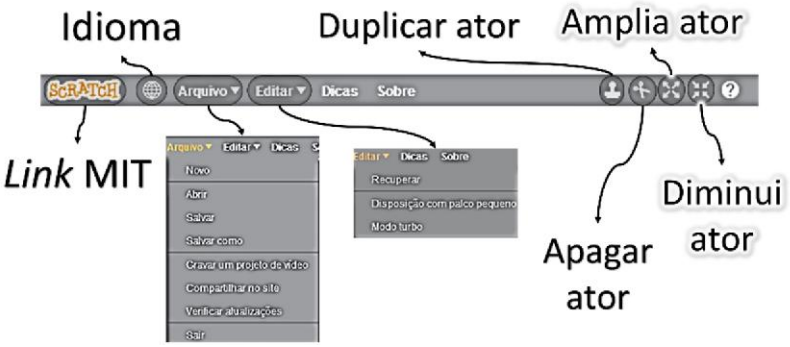
Figura 7 – Alterar o idioma



Barra de menu

Da esquerda para direita, na barra de menu existe: a Logomarca do Scratch é um *link* para o site do MIT. Esse *link* envia para a página inicial do Scratch, por intermédio do navegador; o Globo, que altera o idioma da interface; no menu Arquivo, pode criar um projeto novo, abrir um existente, salvar, gravar um projeto de vídeo (salva pequenos arquivos de 60 segundos no formato de vídeo *flash player*), compartilhar no *site* do Scratch (precisa estar *logado*, ou informa o nome de *login* e senha do site); no menu Editar, pode desfazer uma exclusão de um ator ou blocos (recuperar) ampliar o palco e ativar o modo turbo; Dicas que abrirão um aba no canto direito da tela com *links* para arquivos no site do projeto Scratch e informações dos recursos; O Carimbo duplica um ator no palco (porém, sem movimento ou ação) com todas as informações existentes nesse ator; A Tesoura apaga o ator no palco; As Setas apontadas para o exterior aumentam o tamanho do ator no palco e a setas para o interior diminui o tamanho do ator no palco (Figura 8) .

Figura 8 – Apresentação da barra do menu



A mochila (Figura 6) é uma ferramenta que aparece, quando *logado,* para guardar um bloco montado. Esse recurso é interessante, porém também é possível passar um bloco de um *spriter* para outro, sem esse recurso.

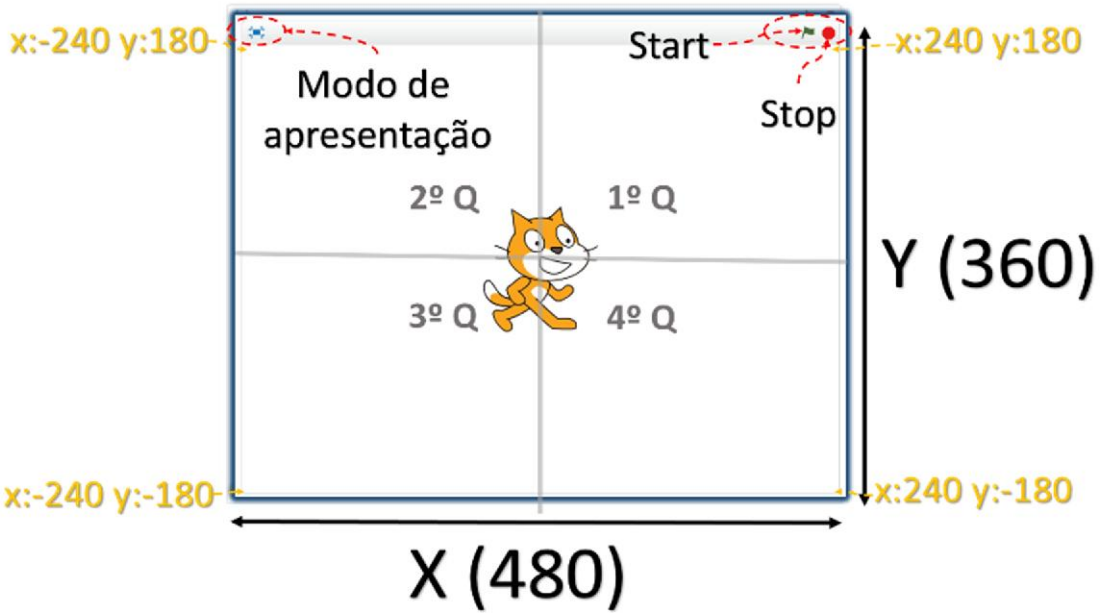
O recurso turbo aumenta a velocidade da execução de alguns blocos. Por exemplo, um bloco mova **[mova (*1000*) passos]** sem o modo turbo dura 70 segundos, já no modo turbo ocorre em 0,2 segundos. O modo turbo também pode ser ativado dando um click na bandeira verde com a tecla *shift* apertada.

Palco

O palco é o local onde ocorrem todos os eventos. Os atores interagem para criar as histórias. No canto superior direito está a maximização do palco (quadrado azul), onde aparece o palco no modo de apresentação. No canto superior esquerdo do palco estão os botões *start* (bandeira verde) e *stop* (octógono bordô).

O palco é graduado em duas dimensões, por isso, cada ator possui uma posição coordenada x e y. O tamanho do palco é de 480 por 360 unidades. São 480 unidades de largura (ordenada x) e 360 unidades de altura (ordenada y).

Figura 9 – Palco



A coordenada (0,0) é o centro do palco. Considerando o par ordenado (0,0) - centro, o palco é um plano cartesiano de quatro quadrantes (Q): no 1º Q as coordenadas variam na diagonal de (0,0) até (240, 180); no 2º Q as coordenadas variam na diagonal de (0,0) até (-240, 180); no 3º Q as coordenadas variam na diagonal de (0,0) até (-240, -180); e no 4ºQ as coordenadas variam na diagonal de (0,0) até (240, -180).

No canto inferior direito do palco aparece a coordenada do cursor do mouse (Figura 10).

Figura 10 – Coordenada do cursor do mouse no palco

Figura 10A – Cursor do mouse no canto superior esquerdo do palco, coordenada (-173,98)

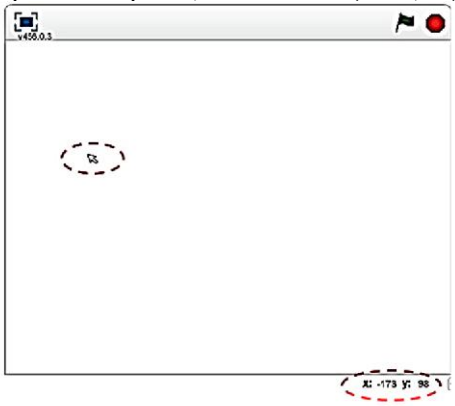
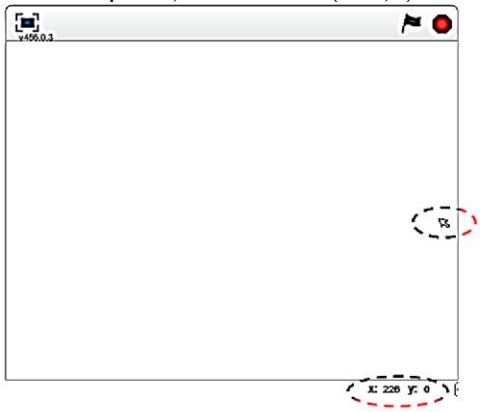


Figura 10B – Cursor do mouse no centro direito do palco, coordenada (226,0)

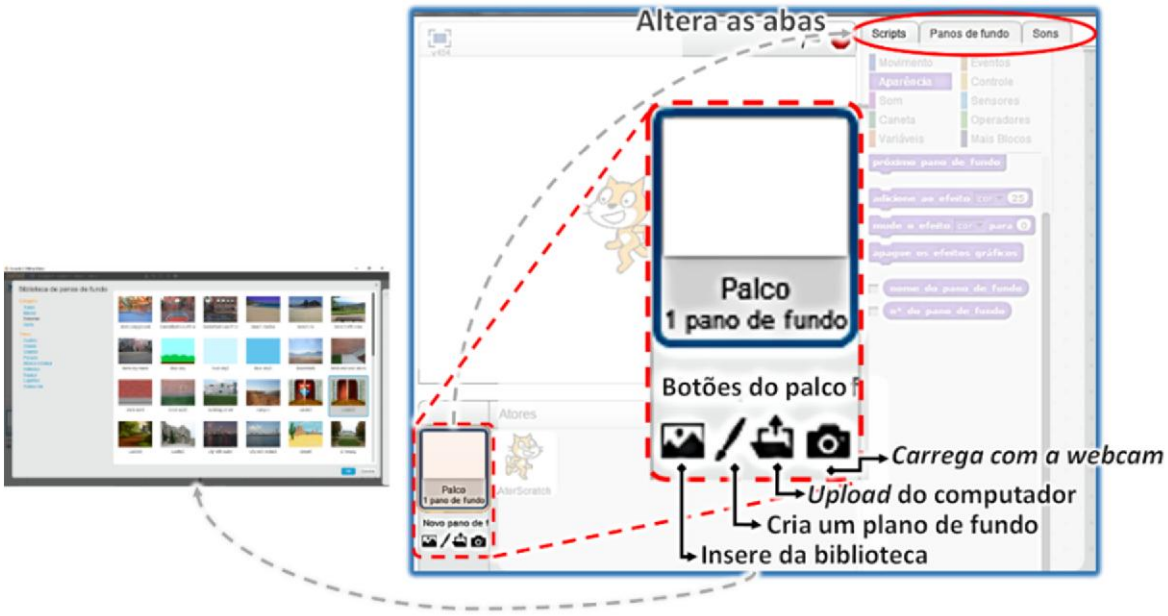


Pano de fundo

A área de apresentação onde ocorre o evento é o palco, essa área pode receber uma imagem que ocupe totalmente ou parcialmente a área, essa imagem é denominada de plano de fundo.

O plano de fundo pode ser alterado durante os eventos, como uma mensagem visual, para que ocorra um evento ou, na decorrência de um dado evento. Por exemplo: considere que um plano de fundo quando aparecer, deve ocultar todos os personagens, ou colocar no palco um novo personagem. Ainda é possível pensar que uma mudança na coloração do palco pode ser uma mensagem de perigo (se em um jogo o personagem estive perdendo vida o palco fica vermelho). Por isso, o palco possui ferramentas específicas para alterar o plano de fundo.

Figura 11 – Criação de plano de fundo do palco



Os planos de fundo podem ser carregados a partir da biblioteca do próprio *software* como a partir do computador ou da *webcam*, mas pode ser criado na própria interface. Quando é selecionado o palco (destaque pontilhado da Figura 11) a aba de fantasia do *spriter* torna-se "plano de fundo" (destaque da elipse na Figura 11) e os blocos passam a conter informações específicas para desenvolver e trabalhar os efeitos do plano de fundo.

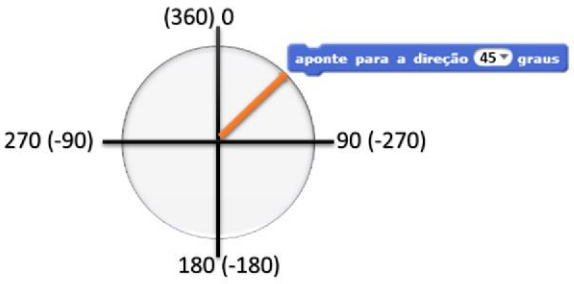
Spríter e ator

Um *spriter* é um desenho, conforme já mencionado, no caso do Scratch é uma imagem bidimensional. Um *spriter* quando disposto no palco é um ator. Assim como o palco, o ator tem ferramentas que permitem carregar um personagem novo a partir da biblioteca, do computador ou da *webcam*. Também é possível criar o personagem. O local onde ficam os personagens, ou *spriters,* é o "banco de *spriter*", que se localiza abaixo do palco.

Na Figura 12 o banco de *spriters* apresenta como acessar as informações. Com o botão direito sobre o "i" no círculo azul do *spriter,* é possível abrir a aba para informações, duplicar, apagar, salvar a imagem no computador ou esconder. Quando se abre a aba de informações, aparece o nome do ator "1", a coordenada "2" e a direção "3" (caso ocorra um movimento). Se o campo "4" for selecionado, fixa a imagem e se o campo "5" for selecionado, o ator aparece no palco, caso contrário fica oculto. A rotação determina o comportamento do ator no palco.

Figura 12 – Banco de *spriters*

A direção do *spriter* é dado pelo círculo da rotação em graus, onde 90º é igual a -270º, considerando a circunferência de 360º no sentido horário, 0º/360º é a parte superior da circunferência. Ou seja, 0º e 360º equivalem a direção do ator para cima, 90º e -270º equivalem a direção para direita, 180º e -180º equivalem a direção para baixo e 270º e -90º equivalem a direção para esquerda.



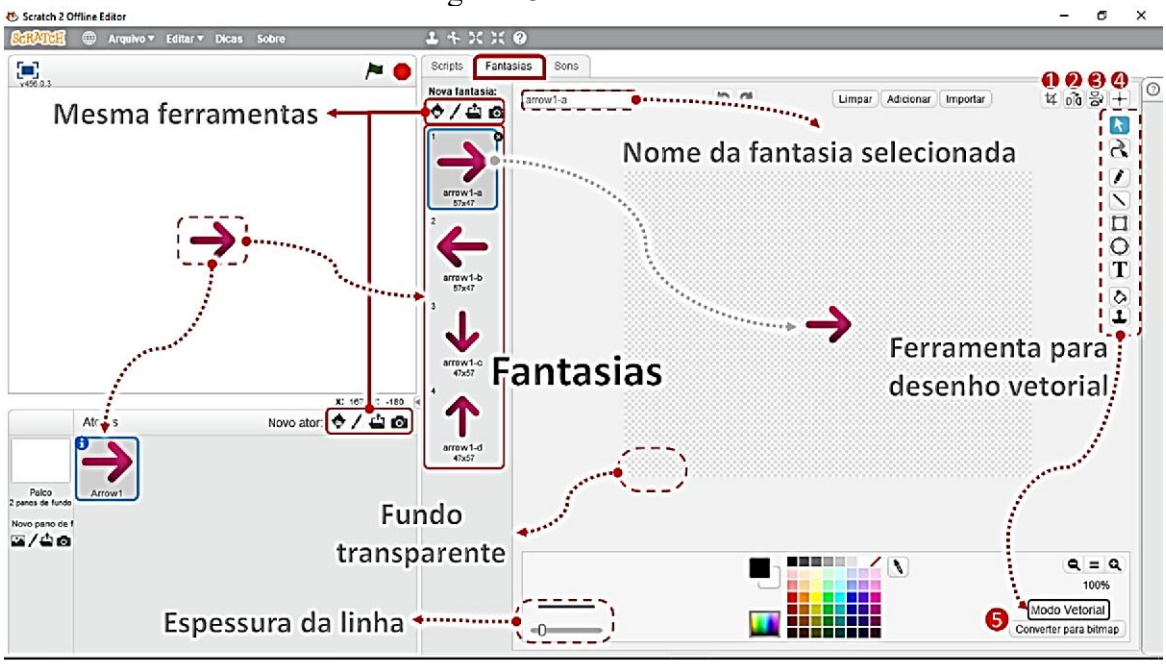
Abas: Fantasia e sons

Observe que assim como o palco, cada *spriter* tem um projeto, um grupo de blocos que produzem as ações e efeitos, sons e fantasias. A aba de *script* será estudada com as ações dos atores.

A aba de fantasias, assim como qualquer aba, apresenta as informações do *script* selecionado. A mudança entre as fantasias pode dar efeito de movimento para o ator. Na Figura 13 é apresentado o ator *arrow*, esse ator possui quatro fantasias (flecha para direita –

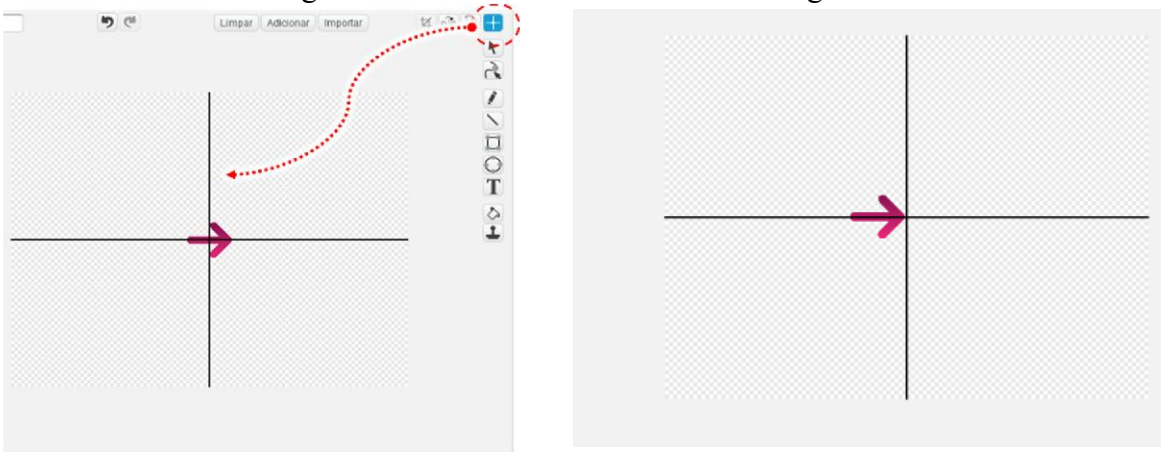
selecionada, flecha para esquerda, flecha para baixo e flecha para cima). O fundo de edição é transparente. No canto superior direito da área de edição aparece o nome da fantasia.

Figura 13 – Aba fantasias



No canto superior direito, aparecem as ferramentas para cortar a imagem (1), para espelhar horizontalmente (2) e espelhar verticalmente (3), o número (4) permite centralizar a imagem. O centro da imagem passa a ser a posição de intercessão entre as duas retas ortogonais. Para mudar o centro da imagem basta mudar a posição das retas ortogonais (Figura 14).

Figura 14 – Deslocamento do centro da figura



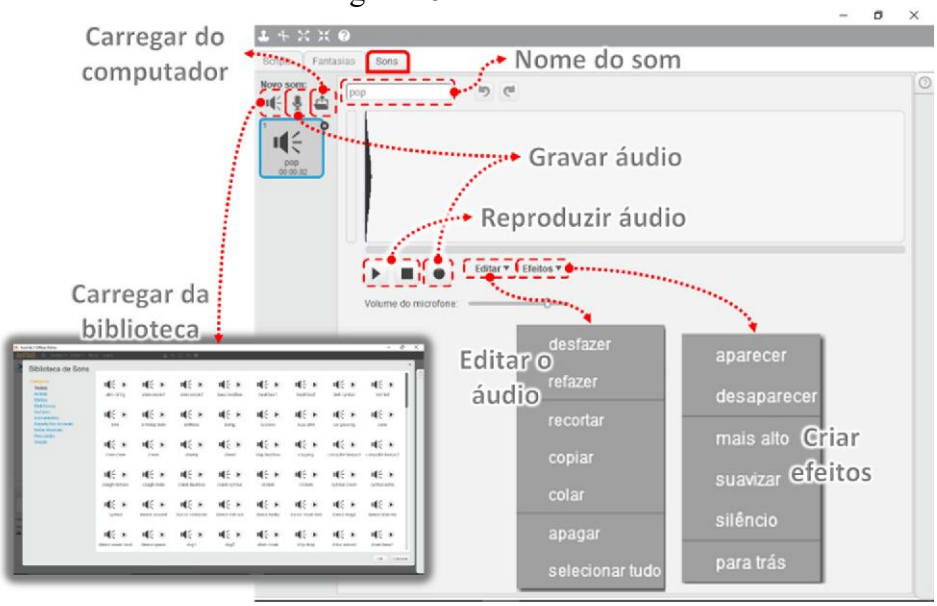
Na Figura 13 o número (5) altera entre o modo vetorial e *bitmap*. São alteradas as ferramentas também, conforme o Quadro 1.

Quadro 1 – Ferramentas para editoração de imagem no Scratch

Modo bitmap, régua na esquerda		Modo vetorial, régua na direita	
	Pincel para colorir	Selecionar objetos	
	Desenhar linha reta	Remodelar linhas por pontos	
	Desenhar retângulos e quadrados	Lápis (mesmo efeito pincel)	
	Desenhar círculos ou forma circulares	Desenhar linha reta (com pontos)	
	Digitar texto	Desenhar retângulos e quadrados (com pontos)	
	Pintar área ou efeito gradiente	Desenhar círculos ou forma circulares (com pontos)	
	Borracha para apagar área	Pintar área ou efeito gradiente	
	Selecionar área retas	Digitar texto	
	Selecionar área para remoção de *backgroud*	Duplicar a imagem	
	Duplicar a imagem		

A aba de sons permite inserir um áudio da biblioteca do próprio *software* ou do computador (apenas arquivos no formato “mp3” e “wav”) e ainda gravar um áudio diretamente no ambiente Scratch.

Figura 15 – Aba sons



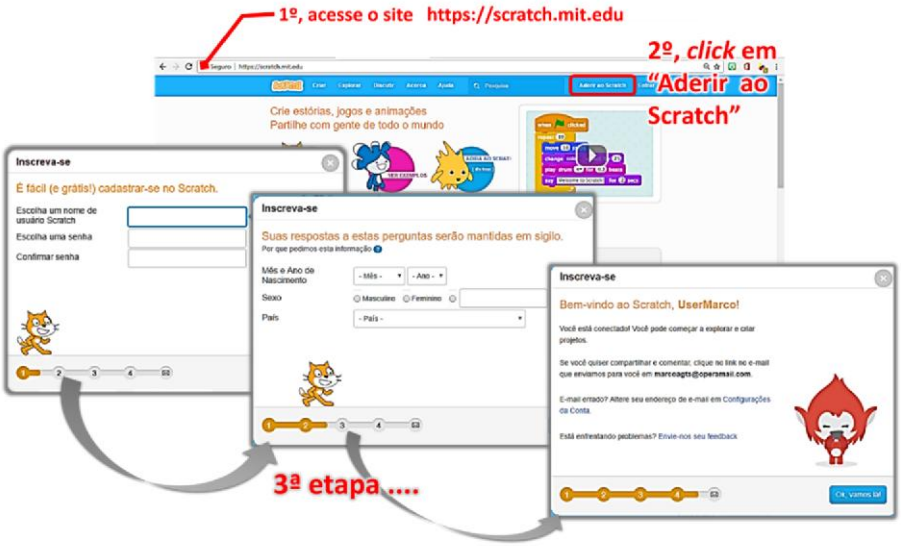
Utilizando os blocos de áudio é possível carregar vários áudios ou definir vários instrumentos (21 instrumentos), conforme é apresentado na Figura 16.

Figura 16 – Definição do instrumentos e nota musical pré-definidos

Para finalizar a apresentação, a área de *script* e os blocos serão vistos conforme forem apresentados os eventos.

Contudo, faltou apresentar como é possível desenvolver o Scratch *on-line*. Inicialmente precisa acessar o site do Scratch (https://scratch.mit.edu) e clicar em Aderir ao Scratch (2º click da Figura 17) e responder as perguntas que aparecerá no *pop-up* (3ª etapa da Figura 17).

Figura 17 – Cadastramento no site



Depois de realizar o cadastro, poderá entrar no Scratch *on-line* e criar, explorar, salvar seus arquivos nas nuvens. Caso tenha desenvolvido um projeto no seu *desktop* e queira compartilhar esse arquivo ou simplesmente fazer uma cópia nas nuvens, basta ir ao <Menu>, da barra de ferramentas principal, </Arquivo /Compartilhar no site> e preencher as informações da Figura 18 (nome do projeto que acha conveniente, *login* e senha que cadastrou no site).

Figura 18 – Aba no Scratch *desktop* para salvar um projeto no site do projeto Scratch

Blocos

Os blocos representam os comandos ou ação diferente, que dizem ao aplicativo como executar. Ou seja, o bloco é a sintaxe do ambiente Scratch. Os blocos, conforme já foi mencionado, são organizados por cores, agrupados pela funcionalidade e dispostos na aba *script*.

Figura 19 – Aba *scripts*

No grupo Movimento estão reunidos os códigos (blocos) que inserem o ator no palco, controlam a posição, direção, rotação e movimento. Os blocos de Aparência definem a fantasia, tamanho e efeitos (cor, olho de peixe, rodamoinho, "pixelização", mosaico, brilho e fantasma), ocultam ou mostram o ator, o plano de fundo e contêm as ferramentas de textos. Os blocos de Som controlam os volumes, a inserção de instrumentos e notas, ou áudios de arquivos. O conjunto de códigos da Caneta permite criar efeitos de desenhos com diferentes cores. Os Eventos que podem interferir são controlados em um bloco especial com topo arredondado e podem monitorar quando se iniciarem as ações entre os atores do projeto, quando uma tecla for acionada, o movimento de vídeo, alteração no plano de fundo, *click* no ator, envio e recebimento de mensagens interna entre os eventos. O grupo de Controle desencadeia funções lógicas que testam uma condição, repetem um evento, esperam um tempo, ou uma condição; nesse grupo, também é possível criar um clone de um ator e controlar. Os blocos dos Sensores detectam o contato com o mouse, da borda do palco, uma cor definida, a posição do mouse, emite uma pergunta e armazena em uma variável, monitora uma tecla, o vídeo, cria cronômetro, posição ou fantasia de um ator. Os blocos Operadores permitem criar relações lógicas entre as quatro operações matemáticas, o resto de uma operação, arredonda um valor, efetua comparação (igual, maior ou menor), condições (e, ou, não), junta caracteres de texto (frases ou palavras), um caractere em determinada posição, o tamanho do grupo e caracteres (palavras) e algumas funções aritméticas (módulo, arredondar, raiz quadrada, seno, cosseno, tangente, arco seno, arco cosseno, arco tangente, ln, log, e^, 10^). As Variáveis são criadas conforme a necessidade da programação. É possível criar uma variável para um propósito ou para uma lista de informações. O Mais Bloco permite adicionar extensão como *PicoBoard* (placa eletrônica) ou bloco Lego, bem como criar um bloco que possa receber entrada numérica, de texto, lógica ou rótulo.

Os blocos, na maioria são parecidos, sendo diferenciados pela cor. Os encaixes são no topo e na base de cada um. Alguns blocos recebem informações (parâmetros). E os blocos de parâmetros ou operadores lógicos não têm encaixe, porém podem também receber parâmetro.

Figura 20 – Formatos dos blocos

Blocos com parâmetros configuráveis | *Blocos com parâmetros semi-configuráveis* | *Blocos parâmetros* | *Blocos operadores*

Alguns blocos não têm o encaixe na parte superior, sendo apresentados num formato semicircular, na parte superior. Em geral, possuem comportamento de monitoramento.

De forma resumida, observam-se cinco formatos de blocos: os empilháveis simples ou blocos de **comando** (são os mais comuns), que possuem uma cavidade (encaixe) na parte superior e na parte inferior. Esses blocos são empilhados um sobre ou sob o outro; há blocos com encaixe apenas na base. A parte superior possui uma área arredondada (como se fosse um chapéu). Esses blocos são utilizados no topo da pilha e monitoram a ocorrência de um evento ou uma mensagem, que são os blocos ***trigger*** (**desencadeador**). Há também os blocos sem encaixe, que são utilizados para passar **parâmetros** para outros blocos, são os blocos de **função**.

Esses blocos são reconhecidos como repórteres ou informantes. E os blocos com formato diferenciado, que criam *loop,* que são blocos de **controle**.

Além do formato, existem outras características, como os blocos que possuem uma lacuna, ou uma área para entrada de dados (parâmetro), que são configuráveis. Esses blocos podem receber também outros blocos como parâmetros. A utilização de blocos e combinação pode ser variável.

Quadro 2 - Blocos de operação de randomização com saídas diferentes

Bloco parâmetro	Faixa de saída
número aleatório entre -5 e 10	{-5, -4, -3, -2, -1, 0, 1, 2, ..., 10}
número aleatório entre 0 e 1.0	{0, 0.1, 0.15, 0.178, 0.2, ..., 1.0}
número aleatório entre 0 e 100 / 100	{0, 0.01, 0.02, 0,03, 0,04, ...,1.0}
10 * número aleatório entre 0 e 10	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...,100}

Observe que diferentes padrões de informação, configurando um mesmo bloco, podem ter diferentes retornos. Esse princípio será aplicado para todos os blocos, bem como os de controle ou comando que permite receber o parâmetro.

Outro tipo de parâmetro é o menu *pull-down* (puxar para baixo ou menu suspenso), que tem o parâmetro pré-estabelecido, conforme demonstrado na Figura 16 (teclas do piano ou lista de instrumentos).

O bloco Variável ou Lista possui um combo-box ao lado, para selecionar se deseja que seja exibido no palco. Caso apareça no palco, o bloco Variável pode apresentar-se em três formatos: só o valor, com painel; apenas o nome e o painel do valor ou com uma barra deslizante para alterar o valor. Tanto o bloco de Variável como o de Lista, ao serem criados, eles criam outros blocos para serem operados e utilizados como parâmetros.

Figura 21 – Blocos especiais

Figura 21A – Bloco Variável

Figura 21B – Bloco Lista

Blocos personalizados

Os blocos da aba "Mais blocos", são blocos especiais, que servem para quebrar algoritmos muito extensos em partes menores, e como são menores, são mais controláveis, caso ocorra erro.

Ao criar um bloco, você informa que tipo de parâmetro ele vai receber, se for o caso, e ainda pode colocar um texto (Figura 22A). Após criar, aparecem o bloco controle **[defina]** como o parâmetro e o bloco de encaixe para ser utilizado. No bloco **[defina]** estão os parâmetros que podem ser arrastados.

Figura 22 – Criação de bloco "Mais bloco" e os parâmetros

Figura 22A – Criar um Bloco

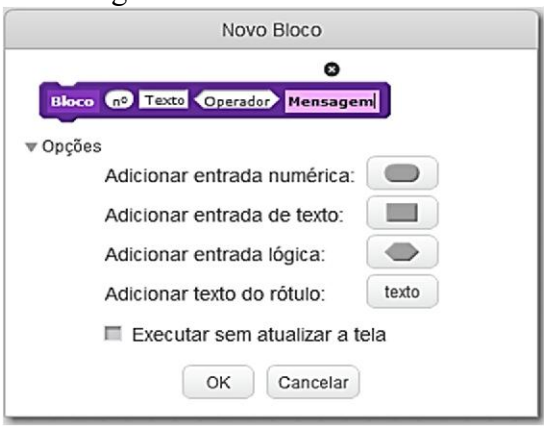
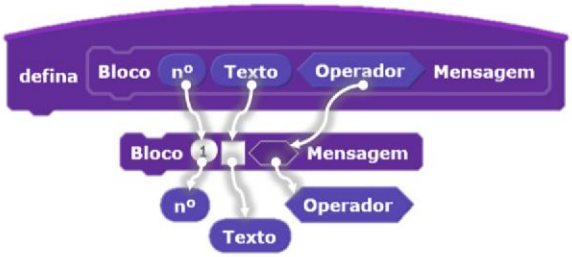


Figura 22B – Bloco Criado



Os blocos de entrada pelo teclado recebem qualquer valor (texto ou número). Ao acionar o bloco **[pergunte ( ) e espere],** é apresentado na parte inferior da tela uma área para criar a informação. A pergunta aparece na tela, acima do ator, como um balão de conversa. A interface Scratch espera até que a tecla "*Enter"* seja pressionada, ou o sinal de "checagem" seja clicado (Figura 23A). Após teclar "*Enter"* ou "checar, o conteúdo digitado é armazenado no bloco **[resposta]**.

Figura 23 – Uso de bloco de entrada pelo teclado

Figura 23A – Iniciando a iteração na tela

Figura 23B – Apresentando a iteração

Os blocos são organizados por cores e estão disponíveis para serem arrastados da paleta de blocos para área de *script*. Cada paleta possui uma série de blocos na mesma cor da paleta (função), o que facilita a identificação do que está acontecendo no algoritmo de determinado projeto.

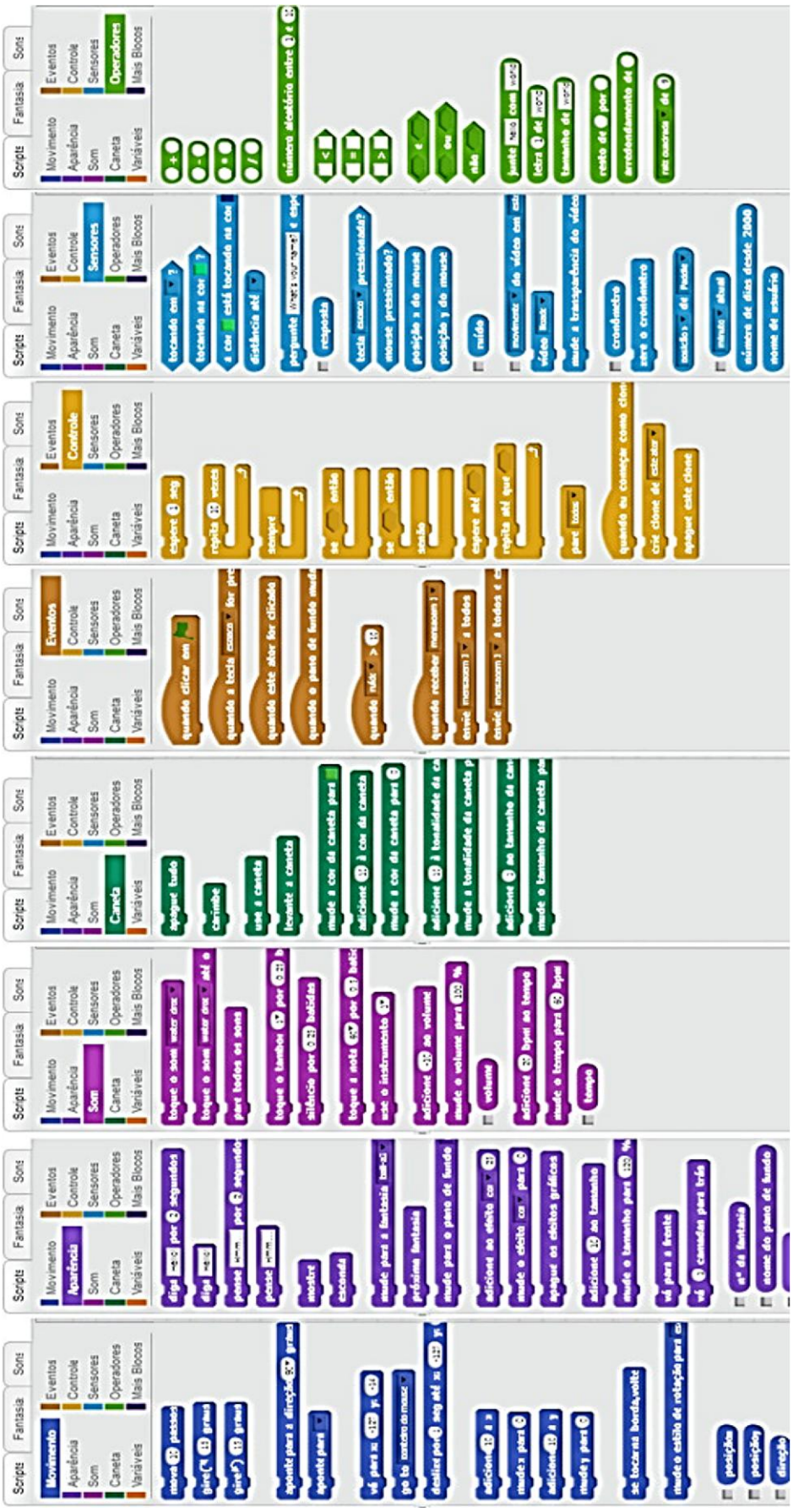
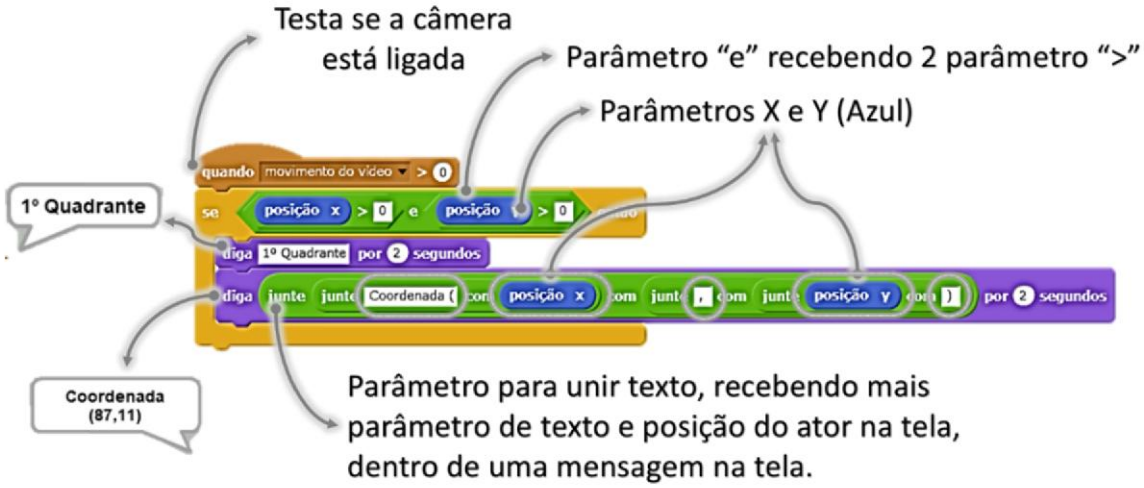


Figura 24 – Aba Scripts com todas as paletas

Algoritmo (script) demonstrativo de combinação de blocos

O algoritmo a seguir usa um monitor de evento para saber se o vídeo está ligado (se o movimento no vídeo for maior que 0) e testa se o ator está no primeiro quadrante, usando dois blocos de lógica “>” e conector “e” (se “x” e “y” maior que zero), envia uma mensagem que é o 1º Quadrante. Usando o mesmo bloco de mensagens, junta mais quatro blocos de concatenação textual para enviar a mensagem e dizer a Coordenada “x” e “y”.

Figura 25 – Uso de bloco e parâmetros



Como é observado, não há limites para combinar os blocos, porém a combinação vai depender apenas de “como” e “para que” se pretende usar. No algoritmo da Figura 21, observa-se dois testes que poderiam ser realizados em uma única etapa, conforme a Figura 22; assim, nota-se que a combinação entre os blocos vai depender de identificar a funcionalidade e aplicá-lo.

Figura 26 – Uso de bloco e parâmetro (sem a condição)

Por que existem três tipos de entradas?

Antes vamos entender outra coisa. Na computação, cada vez que é armazenado algo (informação), é necessário ter um espaço físico para isso. Para entender melhor, considere que você sempre compra 5 kg de arroz e armazena no pote destinado para isso. Por algum motivo, você precisou adquirir 15 Kg. O que vai acontecer? Como você vai armazenar os 10 Kg excedentes?

Agora pense mais um pouco, imagine que por outro motivo você vai precisar substituir o arroz pelo macarrão. 5 Kg de arroz e 5Kg de macarrão podem não ocupar o mesmo espaço, pois pode ser macarrão parafuso, que tem um volume diferenciado, ou pode ser espaguete, que ocupa menos espaço, porém necessita de pote com altura diferenciada. Considere ainda que o cuidado para empilhar o arroz é diferente do macarrão, pois não desejamos que quebre.

No caso do volume excedente (entre 5 e 15 Kg), é provável que seja destinado um lugar temporário, porém que pode incomodar; no caso do volume diferenciado, é provável que deixemos sem tampar. Ou seja, a solução não é a adequada. Mas, na computação isso não é possível. É necessário ter espaço e formato correto, pois uma informação em local errado pode perder parte dele. Assim, na área de sistema cada informação tem seu valor.

Agora vamos entender o Scratch. Existem básicamente três formas, ou seja, temos três tipos de dados: os valores tipo string (caracteres) que são as entradas nos blocos com espaços de canto restos "[ ]"; os valores numéricos (interos e fracionados) que são as entradas nos blocos com cantos arredondados "( )"; e os que recebem valores lógicos ou booleanos, que são blocos com formato de "hexágono irregular – <>".

Note que alguns blocos também possuem formatos diferenciados como as entradas de dados e, estão relacionados com a mesma lógica.

Discutindo temas para trabalhar

"*Quando o educador ultrapassa o posto de mero reprodutor de conhecimento, assumindo a postura de transformador da realidade, enxerga a importância da forma de condução do ensino dos alunos, independentemente da etapa de escolarização. Assim, não é possível promover a transformação com práticas de ensino engessadas e baseadas na repetição para a memorização. Tais métodos assumem uma postura domesticadora de educação, tantas vezes criticada por Paulo Freire.*" [1]

É importante refletir sobre a reprodução de conhecimento, principalmente quando temos temas que poderíamos (ou deveríamos) desenvolver além do contexto dos livros didáticos, para facilitar o entendimento do aluno. Nesse fórum queremos discutir esses conteúdos.

[1] (COSTA, Jaqueline de Morais; PINHEIRO, Nilcéia Aparecida Maciel. O ENSINO POR MEIO DE TEMAS-GERADORES: a educação pensada de forma contextualizada, problematizada e interdisciplinar. **Imagens da Educação,** Londrina, v. 2, n. 3, p.33-44, dez. 2013)

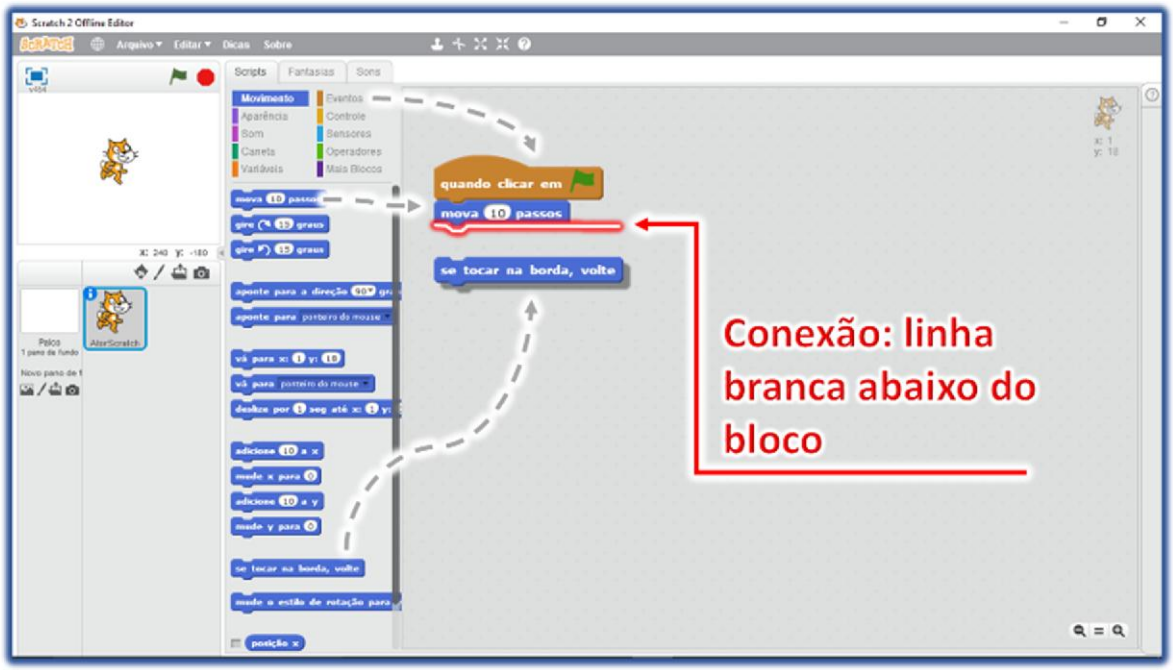
2º Tópico: Iniciando as atividades no Scratch

A finalidade dessa etapa é apresentar e ambientar para que o participante identifique as funções da interface de programação, criando seu primeiro projeto simples – Usar um spriter pronto, dar movimento e controlar.

1º projeto – *script* inicial para criar eventos

Para criar um *script,* só é necessário arrastar o bloco das paletas para a área de *script* (Figura 27). Advertimos que a criação é simples e fácil, mas, para que o evento aconteça como esperado, é necessário existir uma lógica.

Figura 27 – Montando o *script*

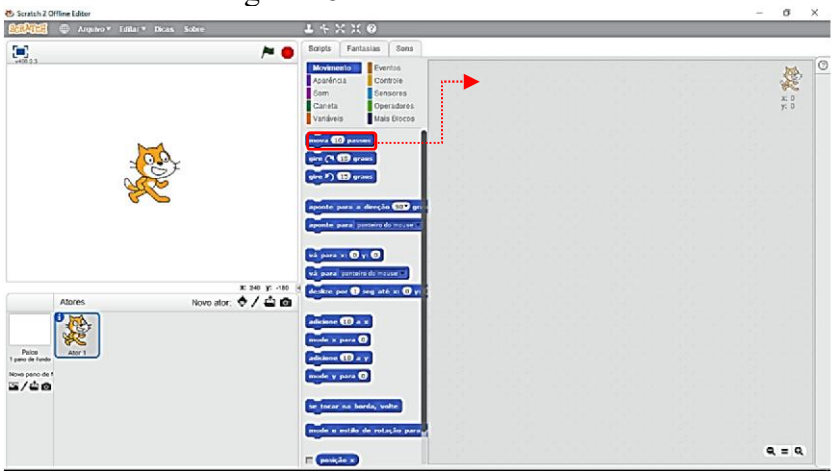


Observe que na Figura 27 aparece uma linha branca (na imagem destacada com a cor vermelha) abaixo do bloco que vai receber o novo bloco.

Criando movimento

Então, para iniciar, vamos criar um primeiro *script* para o gato (ator padrão do Scratch) se movimentar no palco. Ao iniciar a interface do Scratch, o gato aparece no meio do palco, Coordenada (0,0) e a aba de movimento fica selecionada.

Figura 28 – Arrastando o bloco



Na aba Movimento, arraste **[mova (*10*) passos]** para a área de *script*. Se der um *click* nesse bloco, o gato se movimentará para a direita, pois a informação padrão que está nele (ator) é direção 90º (que indica direita). Se continuar clicando, ele continuará se deslocando.

Porém, essa não é a tarefa que pretendemos, a repetição é uma condição que a computação resolveu bem. Então, na aba Controle, arraste o bloco de *loop* **[sempre]** e coloque o **[mova (*10*) passos]** dentro. Observe que ao clicar, o gato vai sair do palco, fincando uma pequena parte aparecendo.

Precisamos fazer com que ele retorne para tela. Então, volte à paleta de Movimento e arraste o bloco **[se trocar na borda, volte]** para dentro do *loop* **[sempre]**.

Se forem realizados todos os passos, ao clicar no bloco, observaremos que o gato vai e volta, mas volta de cabeça para baixo. Isso poderemos resolver também.

Para resolver a posição do gato no palco, na paleta Movimento tem um bloco **[mude o estilo de rotação <*esquerda-direita*>]**, arraste esse bloco para dentro do *loop* **[sempre]**, abaixo do tocar na borda.

O movimento no Scratch pode ser relativo ou absoluto, ambos os movimentos são desenvolvidos pelo centro do ator. Movimento absoluto determina o destino do ator (por exemplo: vá para x,y), o movimento relativo está relacionado aos deslocamentos que definem o movimento a partir da origem (por exemplo: mova 10 passos) e podem estar relacionados ao deslocamento no palco, a direção e ao estilo de rotação (relembrando a rotação é em graus, onde 90º é igual a -270º, considerando a circunferência de 360º no sentido horário, onde 0º/360º é a parte superior da circunferência).

Inserindo um áudio

Para inserir um áudio no nosso ator, vamos criar um controle e um sensor. Arraste o bloco **[se < > então]** da aba Controle para dentro do *loop* **[sempre]**. No espaço do parâmetro inclua o bloco Sensor **[tocando em <*borda*>?]**#*mude o parâmetro de "ponteiro do mouse" para "borda"*#. Dentro desse *loop* de controle insira o bloco **[toque o som <*miau*>]**. Clique no *script* e veja o que acontece.

Mudando a fantasia

O gato agora vai e vem, mia ao tocar na borda, porém não mexe as pernas, observou isso? Vamos resolver? Vá até a aba Fantasia e coloque **[próxima fantasia]** dentro do *loop* **[sempre]**.

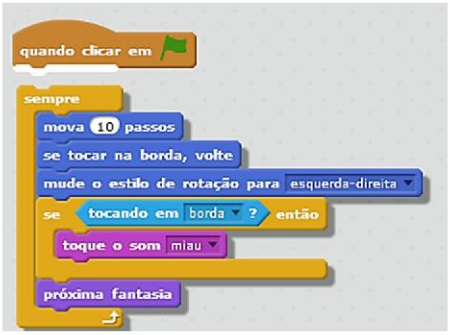
O que está faltando? O monitor de eventos de quando começar (clique na bandeira) para que ele faça sozinho. Então, vá até a aba Eventos e arraste **[quando clicar em ⚑]** e coloque no

topo do bloco que foi criado. Pode ser que não consiga, ou tenha alguma dificuldade, mas é simples. Deixe o bloco na área de *script* e arraste todo o bloco que foi construído para baixo desse bloco de Evento. Parece ser mais fácil dessa forma, mas é porque o Scratch está preparado para receber um bloco sempre abaixo (Figura 29B). Caso tenha conseguido, de cima para baixo, observe que formou uma imagem (silhueta) do bloco no encaixe (Figura 29A).

Figura 29 – Encaixe dos blocos

Figura 29A - Encaixe de cima para baixo

Figura 29B - Encaixe de baixo para cima



Analisando o 1º *projeto*

Observamos nesse primeiro *script*, que além de conhecer o bloco, devemos ter atenção para os efeitos desses blocos: (i) para andar, devemos ter um limite até onde, no caso, até a borda do palco; (ii) se o efeito é para voltar, como é, e o que acontece? Esse evento mudou a posição, então devemos garantir a volta com alguma lógica, que no caso do *script* foi: se voltar mude o estilo de rotação; (iii) quero um som, mas onde? O *spriter* da biblioteca do Scratch tem seus arquivos personalizados, por isso foi simples aplicar o som "miau"; (iv) o que poderia dar um aspecto mais interessante? Nesse exemplo, foi só colocar o bloco para uma próxima fantasia. Ainda vimos como repetir, sem ter que estar clicando. Ou seja, automatizamos o processo para que o sistema computacional resolva sozinho os movimentos repetitivos.

Observe também que já utilizamos seis abas, (Figura 30B).

Figura 30 – 1º *Script*.

Figura 30A –Ação no Palco

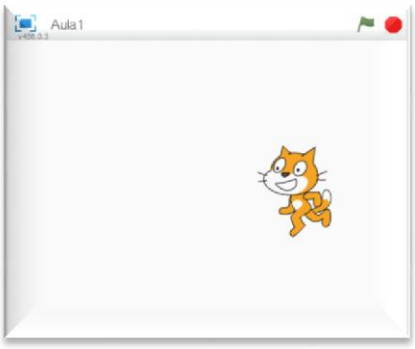
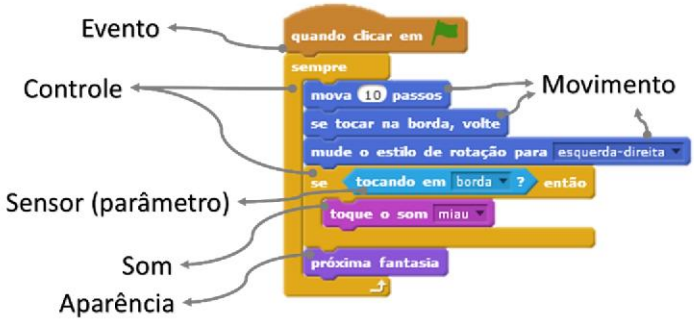


Figura 30B - Análise do *script*



Ao observar a figura acima o ator (gato) parou de frente para o lado esquerdo, mas, quando começou ele estava voltado para a direita e no centro do palco. Como garantir que ele volte sempre para onde estava? Essa é a primeira atividade, iniciar o ator na posição desejada

sempre. Então, você deve fazer com que o ator fique na coordenada (0,0) e com a frente para lado direito do palco de quem observa.

Vídeo deste projeto

https://youtu.be/38gOgoD4qXo

3º Tópico: Ambiente Virtual

A finalidade dessa etapa é identificar recursos para criar um jogo com interação virtual (webcam), criar bloco variável, criar personagens e produzir interações entres os personagens (mensagens).

2º projeto: virtualidade aumentada - usando eventos e parâmetros

Até aqui viemos lentamente. Quando falei em um bloco, disse qual era o caminho (paleta). A partir desse ponto vamos falar apenas no uso de um bloco.

Nessa nova interação, vamos criar um arquivo que determine parâmetros para os demais atores e também vamos criar atores. Esse projeto é para aprender a trabalhar com a câmera (sistema de imersão), viabilizando uma interação do usuário ao ambiente virtual.

Nesse novo projeto, vamos conduzir o ator na direção do movimento da câmera. Para isso, vamos passar o parâmetro para que aconteça esse monitoramento. Criar um objeto que dê uma motivação (jogo) e trabalhar com criação de variável.

Primeiro quero garantir o tamanho e o local do meu ator, então vamos usar o bloco **vá para x:( ) y:( )**]*#nesse bloco vamos preencher com a coordenada (0, 0), garantindo a posição#*, depois vou definir que o ator sempre vá para imagem em movimento no vídeo, mas eu quero mandar um efeito de deslocamento, então vou usar o bloco **[deslize por ( ) seg até x:( ) y:( )**]*#nesse bloco vamos dar um tempo de meio segundo (0.5, passar um parâmetro para preencher x: e valor para y: (parâmetro, -135), garantindo que haverá um movimento#*. O parâmetro para vídeo é o bloco de sensor de **[<*direção*> do vídeo em <*esse ator*>]***#mude de movimento para direção#,* que fará o ator acompanhar o movimento no vídeo, e o valor de y: (-135) é a posição que eu quero que o ator permaneça, no plano cartesiano. Esse bloco fica dentro do *loop* **[sempre]**, garantindo o movimento contínuo. No bloco do sensor, é necessário alterar o "*movimento*" para "*direção*". Podemos já observar que está funcionando. Mas antes vamos automatizar o processo com o bloco **[quando clicar em]**, agora temos a figura a seguir:

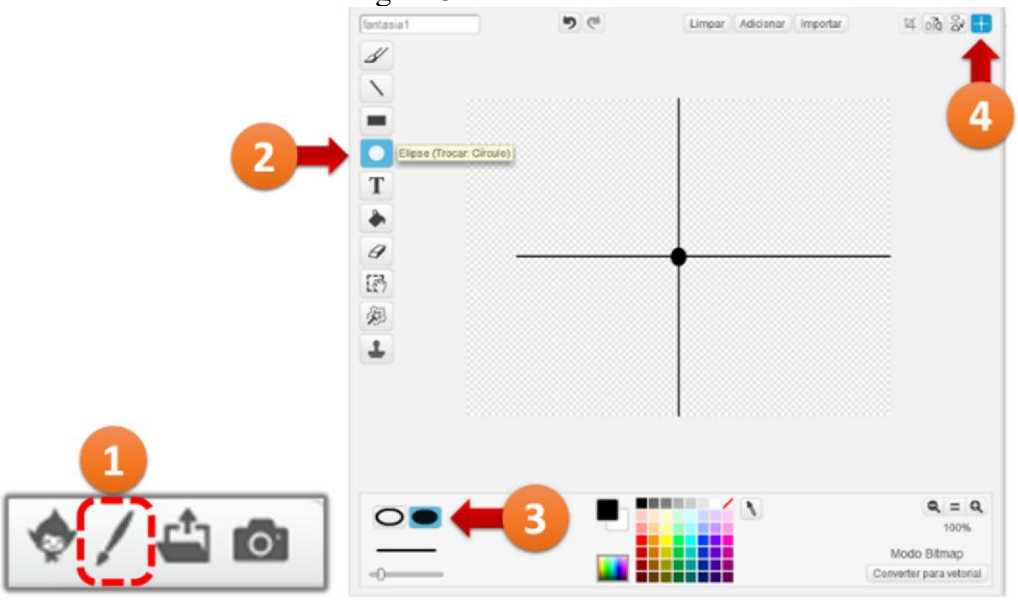
Figura 31 – Parâmetro bloco de vídeo

Vimos o movimento, logo podemos colocar um bloco para alterar a fantasia, igual no primeiro *script* **[próxima fantasia]**.

Agora vamos introduzir outro ator, porém vamos criar um simples. Observe a Figura 32 e a numeração destacada nas circunferências. Ao clicar no pincel (1), vai aparecer a área para criação do ator. Nesse ambiente selecione a Elipse (2) e o formato cheio (3). Desenhe um

pequeno círculo e para garantir que essa imagem (ator) está no centro use a ferramenta (4) e coloque a interseção no centro da circunferência.

Figura 32 – Criar um ator

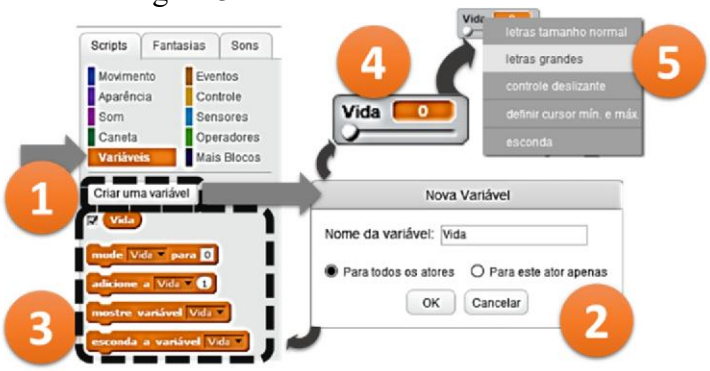


Após criar o ator, clique na aba *script* e vamos fazer esse ator se movimentar no palco, lembrando que temos as bordas de limite. Use o bloco **[quando clicar em]**, bloco **[sempre]**, dentro de **[mova (*10*) passos]** e o controle do limite da borda. Para esse controle, vamos mudar a direção do ator no palco, então vamos usar um bloco **[se < > então]**. Nesse bloco temos que passar um parâmetro, vamos usar um bloco **[tocando em <*borda*>]***#esse bloco é um sensor que está pronto para tocando em ponteiro do mouse, logo é necessário alterar para borda#*. Dentro desse bloco vamos colocar o movimento do bloco **[gire para direita ( ) graus]***#esse bloco de movimento recebe outro parâmetro. Vamos passar como parâmetro o bloco de operadores **[número aleatório entre ( ) e ( )]**, use o valor de "30" e "60"#.*

Figura 33 – Parâmetros blocos operadores

Pronto, temos o ator do gato seguindo a direção do movimento do vídeo e a bola movimento pelo palco. Agora, vamos criar uma pontuação e fazer com que cada vez que os atores se encontrem diminua uma vida do personagem, para isso vamos selecionar novamente o ator 1 (gato) e seguindo os passos da Figura 34 a seguir, utilizaremos a paleta Variáveis para criar uma variável (1) dando o nome de Vida *#sugestão#*, marcar para todos e clicar em "ok" (2). Aparecerá o bloco **[Vida]** como parâmetro e os blocos para controlar esse parâmetro (3).

Figura 34 – Criar um bloco variável



No palco, vai aparecer a variável com controle deslizante (4), para esse *script*. Usaremos letras grandes (5). Para isso é só usar o mouse, botão direito e selecionar "letras grandes".

Então, no *script* do ator 1 (gato) vamos usar o bloco **[Vida]**, pegue um bloco **[se < > então]** e passe como parâmetro para esse bloco o sensor **[tocando em <>]***#mude para Ator1#*. Dentro desse controle use o bloco **[adicione a Vida ( )]***#passe como parâmetro "-1"#*.

Qual é a lógica que estamos criando? Já pensou? Simples, cada vez que os dois atores se encontrarem vai diminuir um ponto variável Vida. Então temos que ter um limite?

Vamos criar esse limite, use o bloco **[se < > então]**, abaixo do anterior e passe como parâmetro para esse bloco o Operador de comparação "menor"**[( ) < ( )]** *#passe como parâmetros a variável [Vida] e o valor 1 #*. Dentro desse controle, vamos enviar uma mensagem. Em Eventos, use o bloco **[envie <nova mensagem...> a todos]** *#clique na nova mensagem, na janela escreva "GameOver" e deixe essa mensagem selecionada#.*

Figura 35 – Criando uma mensagem

Pronto, temos um evento para diminuir o valor da variável "Vida". Porém, quando é que demos algum valor para a variável "Vida"? E, para que serve essa mensagem "GameOver"?

A variável "Vida" está meio que claro, será um processo de contabilização para ver a pontuação. A mensagem é uma forma de enviar um no parâmetro ou informação para todos os atores.

Passando uma mensagem e uso de variável

Em geral devemos inicializar todos os valores das variáveis e sempre que possível em um único local. Então, primeiro vamos criar a pontuação (inicializar) da variável "Vida".

Clique em palco. Arraste o bloco **[quando clicar em ⚑]**, vá em variáveis e arraste o bloco **[mude <Vida> para ( )]***#observe que é "**mude**" não "**adicione**" como o anterior e passe como parâmetro o valor "3"#*.

Vamos dar um efeito legal? Observando a Figura 36, siga as etapas: ainda no palco (1), vá até a aba "Panos de fundo" (2), use a ferramenta Texto (4) e no meio do plano de fundo digite "*Game Over*". Para essa etapa sugiro alterar para o Modo Vetorial (5).

Figura 36 – Criando um plano de fundo

Com a ferramenta texto, no modo vetorial, é possível esticar o texto para o tamanho e posição que deseja.

Vamos garantir que essa mensagem seja vista de forma coerente? Coloque um bloco de Eventos **[quando receber < >]**#*passe como parâmetro a mensagem "Game Over"*#, abaixo coloque o bloco de sensor **[vídeo <desligado>]**.

Agora um detalhe: ao desligarmos, temos que garantir que vai ligar quando iniciar. Então, vá até a tecla *script* **[quando clicar em]** e adicione o sensor **[vídeo <ligado>]**. Observe que com esses eventos criados, será garantindo que os eventos vão ocorrer como e quando desejamos.

Figura 37 – Controlando os parâmetros dos blocos variáveis e eventos

Por falar em organização, quando nosso jogo termina ainda fica a bolinha na tela. Podemos retirá-la, como um simples bloco **[quando receber <*GameOver*>]** e o bloco **[esconda]** e relembrando, devo garantir que apareça, então a primeira coisa do *script* da bolinha, após o bloco da bandeira é o bloco **[mostre]**.

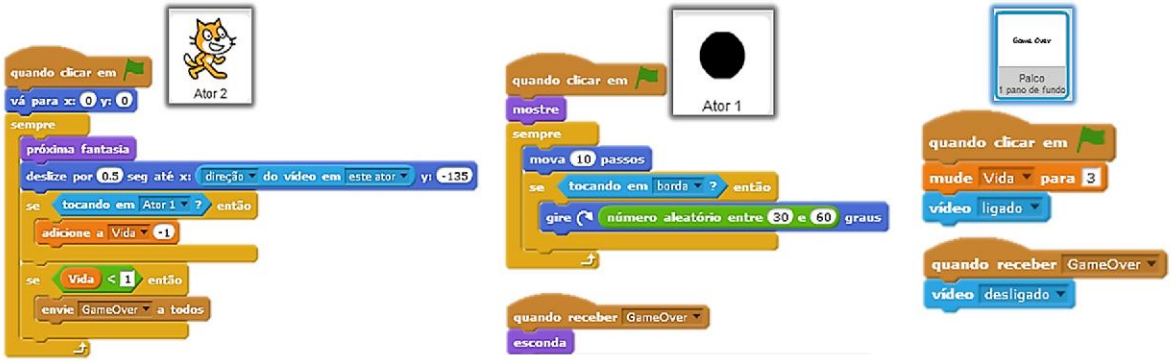
Analisado o script

Então o que fizemos aqui? Criamos um jogo com uma interação entre o real e o virtual, ou seja, usando o sensorial humano para criar ações no ambiente virtual. Na definição do Professor Claudio Kirner, estamos criando uma virtualidade aumentada, em que a "realidade virtual é uma forma das pessoas visualizarem, manipularem e interagirem com computadores e dados extremamente complexos" (KIRNER, 1997). Ainda segundo Kirner, (1997) um sistema que permita utilizar além do fator visual, a captura automática da pessoa e de movimentos, pelo sistema computacional interagindo com o meio virtual é uma técnica de imersão virtual.

Assim, com um pouco de imaginação, temos recursos para criar a virtualidade aumentada ou a tecnologia de imersão para a sala de aula.

E sobre nosso projeto? Vamos dar uma olhada geral nos blocos para tirar dúvidas. Verifiquem se o seu está assim como a Figura 38.

Figura 38 – Controlando os parâmetros dos blocos variáveis e eventos



Script do ator gato | *Script* do ator bola | *Script* do palco

Bem, como sempre temos uma tarefa, mas vamos primeiro entender os problemas. Fazer o jogo de imersão começou a complicar, pois atuamos com mais de um ator, porém é importante refinar ainda mais esse projeto. O tamanho do gato, em relação ao palco e à bola, está grande, pois a relação de movimento é um pouco lenta; isso pode criar eventuais toques dos atores no palco. Então, **sem** usar a ferramenta para reduzir (), e garantindo que estará sempre no mesmo tamanho, diminua o tamanho do gato (sugiro passar para 70%).

Uma questão é que a bola pode vir em um ângulo baixo e por isso iria tocar no gato, por isso, sugiro fazer o gato dar um pulo, com uma ação do teclado.

Outro ponto é que ao iniciar o vídeo a imagem captada fica um pouco fosca, então devemos colocar a transparência para 100% e temos dois panos de fundo. Revendo nossa atividade: fazer esse gato dar um salto e retornar à posição, diminuir o tamanho do gato e mudar a transparência da imagem captada pela webcam vídeo e o pano de fundo.

Vídeo deste projeto

https://youtu.be/m9ffSFv1Ehk

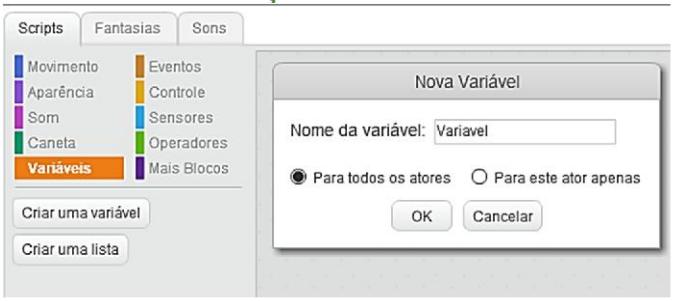
O que é uma variável para a área de computação?

Na programação, uma variável é um objeto capaz armazenar uma informação.

No Scratch, a variável é um bloco capaz de armazenar valores do tipo booleano, número e string.

Após criar o bloco variável, surgem na aba o bloco e as instruções para: definir valor; alterar o valor; ocultar ou mostrar uma variável. No palco, aparece a Variável com nome e o valor. Caso não queira que ela fique no palco, pode desmarcar o check-box, ao lado do bloco com o nome da variável.

Criação da variável



Blocos variável e instruções

Na criação da variável é comum definir quem tem acesso, no Scratch é só marcar se para aquele spriter (ator) ou para todos #aconselho deixar marcado sempre para todos#.

Outro ponto importante é o nome. A regra geral para criar nomes de variáveis é: primeiro, caráter maiúscula, seguida pelos demais minúsculos, sem utilização de acentuação gráfica e caracteres especiais (cedilha, cifrão, arrouba, espaço, números ...). Se o nome for composto por duas palavras, como por exemplo: 1ª variável, o nome fica "PrimeiraVariavel" (caso prefira para ficar curto pode ser "PrimVar"), sem acento, espaço, número ou acento e a segunda palavra inicia-se pelo caráter maiúsculo.

Embora a versão atual do Scratch aceita qualquer formato com letra, números e acentos, sugerimos não utilizar, pois a maioria das sintaxes de programação não aceitam.

O valor da Variável pode ser exibido no palco como monitor, exibindo o nome e valor ou como painel só com o valor e como forma de alteração do valor, como um botão deslizante.

4º Tópico: Efeitos com a caneta

A finalidade dessa etapa identificar os recursos da caneta, do efeito de "loop," duplicar spliter e controlar variável.

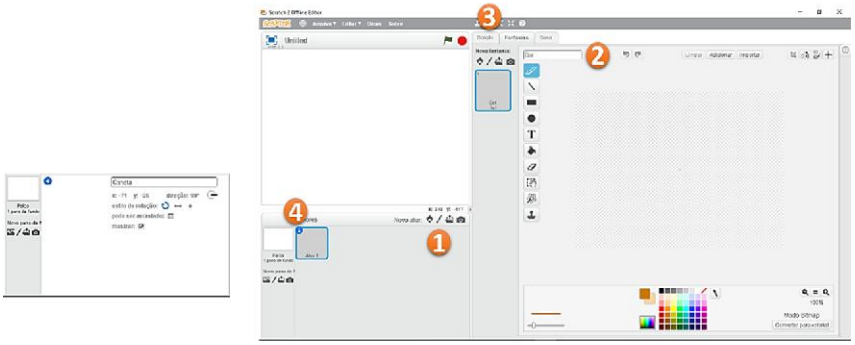
3º projeto: usando caneta, efeitos e interação

Para esse projeto, abra um arquivo novo e apague o *spriter:* Ator 1 (Gato). Use o botão direito no *spriter* e apagar. Após apagar, não é possível criar script. Então, observando a Figura 39, vamos ao pincel (1) criar um *spriter* sem fantasia. Observe que é para criar um spriter sem ator, isso mesmo, não vou dar nenhum *click* na área de criação. Vou nomear a "fantasia nula" (3) de "Dot" (ponto em inglês) (2).

Clicando no "i" do *spriter* (4) vou nomear o *spriter* de caneta.

Observem que não criei nenhum outro ator ou fantasia, apenas criei um *spriter* para poder criar um *script.*

Figura 39 – Criando ator *Dot*



A pergunta é: se não tem ator, o que irá interagir no palco? A resposta é: apenas a caneta.

Então vamos criar esse *script.* Para entender, inicialmente arraste o bloco **[quando clicar em ⚑]**, abaixo dele coloque o bloco **[apague tudo]** (aba de canetas). Isso vai garantir que o nosso *script* comece sempre em um ambiente limpo. Ainda garantindo que não dará erro, vamos levantar a caneta e posicionar a caneta com os blocos **[levante a caneta]**, **[vá para x:( ) y:( )]***#sugiro a coordenada (0,0)#* e o bloco**[aponte para a direção ( ) graus]***#sugiro o valor 90#*.

Agora vamos escolher a cor com o bloco **[mude a cor da caneta para ( )]***#pode ser dois tipos com: cor ou valor#* (Figura 40).

Figura 40 – Blocos controladores de cor

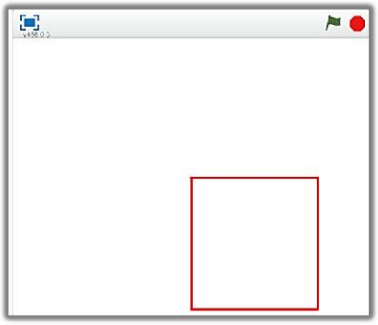
Vamos usar o bloco numérico, isso pode favorecer uma interação. Seguindo, vamos colocar o bloco **[mude o tamanho da caneta para ( )]**. Já que posicionamos, definimos a cor

e o padrão da caneta que queremos, é só usar o bloco **[use a caneta]**e mandar desenvolver o *script*.

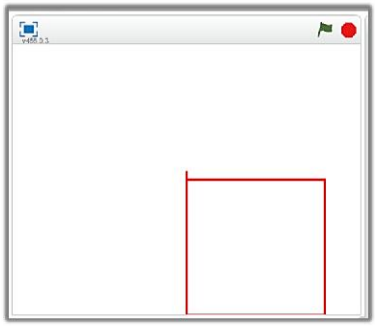
Vamos criar um polígono regular, com lados e ângulos iguais. O mais simples é um polígono de quatro lados, ou quadrado. Se vamos criar um polígono de quatro lados e, os lados são iguais, vamos fazer a mesma coisa quatro vezes, então vamos fazer com o bloco para criar o *loop*: **[repita ( ) vezes]** *#passaremos como parâmetros o valor "4"#*. Um quadrado possui lados e ângulos iguais (regular) então devemos mover a caneta com os mesmos passos para ambos os lados. Por isso, dentro do *loop* vamos colocar o bloco **[mova ( ) passos]** e a seguir um bloco para fazer que ocorra uma mudança na direção. Para isso vamos usar o bloco **[gire "seta direita" ( ) graus]** *#como é um quadrado vamos passar o valor 90º#*

Agora dê um valor qualquer para o bloco **[mova ( ) passos]** e verifique. Observe que se o valor for até 170 passos o quadrado não dá erro, mas se passar de 170 começa a apresentar problemas (Figura 41).

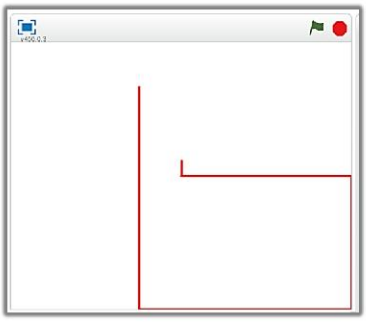
Figura 41 – *Script* do polígono



Polígono com até 170 passos



Polígono com 190 passos



Polígono com 300 passos

Então, vamos criar uma interação como mencionei, só que limitando essa questão do tamanho do polígono.

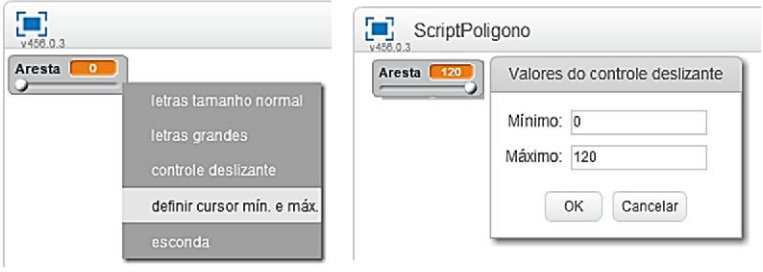
Vamos criar uma variável "Aresta". Após criar, coloque no bloco **[mova ( ) passos]** passando o bloco **[Aresta]** como parâmetro. Note que não me preocupei em dar um valor para variável inicialmente.

Como foi observado, devemos corrigir o valor, para que não passe de 170. Na variável (aparente no palco), mude para "controle deslizante": com o botão direito do mouse sobre a variável. Depois, também como o botão direito, "definir curso mín. e máx." e passe o valor de 0 até 120 (Figura 42), garantindo que o valor não apresentará um problema e, que qualquer valor na faixa escolhida atende e, para esse caso, como foi definida uma faixa de valores, não há necessidade de inicializar a variável com um valor qualquer.

Figura 42 – Controle deslizante

Figura 42A – Controle deslizante

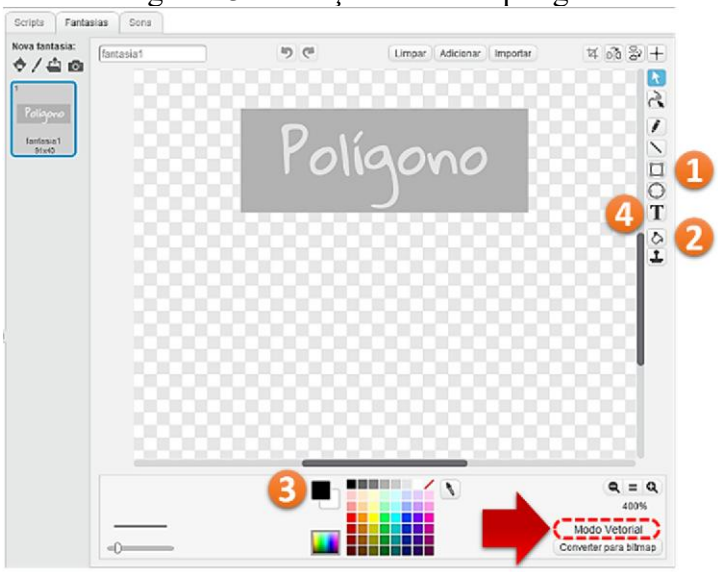
Figura 42B – Definir Max e Min



Ficou bom? Porém, não achei o resultado interessante. Vamos deixar isso melhor?

Vamos criar um "botão" com o nome polígono. Observe a Figura 43, inicialmente sugiro a mudança para o modo vetorial (seta em destaque na figura). Para criar o botão, use a ferramenta retângulo (passo 1) e preencha com a cor que quiser (sugiro a cor cinza), passos (2) e (3) da figura. Coloque o texto "Polígono" no centro da figura (passo 4). Observe que alguns tipos de fonte não aceitam acentuação gráfica. Se isso aconteceu com você, mude a fonte (estou usando a fonte Glória).

Figura 43 – Criação do botão polígono



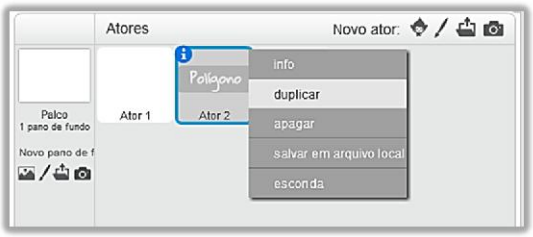
Para o *script,* vamos garantir a posição com o bloco **[quando clicar em ⚑]** e abaixo o bloco **[sempre]**, dentro do *loop* sempre o bloco **[vá para x:( ) y:( )]** *#estou sugerindo os valores -170 e 157, para ficar no canto superior esquerdo, de quem observa a tela#.*

Para o outro *script* simples, criar um com o bloco de controle **[quando este ator for clicado]** e abaixo o bloco **[envie <*nova mensagem...*> a todos]***#crie o parâmetro "Polígono" para o lugar de "nova mensagem..."#.*

Figura 44 – *Script* de criação do *spriter* para o botão Polígono

Agora vamos criar um segundo botão, porém dá para ser mais simples, pois já temos um pronto, então vamos copiar o *spriter* inteiro, já que vai ter a mesma função: só usar o botão direito do *mouse*.

Figura 45 – Duplicação do *spriter* e códigos



Após duplicar, basta ir até fantasia e na ferramenta "texto", alterar o valor de "Polígono" para "Arranjo". Depois no *script* mudar o valor da posição de "x:", no botão polígono eu coloquei "-170", para o tamanho de botão que criei diminui 100. Ou seja, passei o valor de "x:" para "-70" e mantive o valor de "y:", mudando a posição do botão. No bloco **[envie <nova mensagem...> a todos]** *#criei o parâmetro "Arranjo"#*.

Agora temos que fazer as mudanças *script* inicial (*spriter* sem fantasia). Para o *script* do polígono está pronto, vamos apenas arrumar.

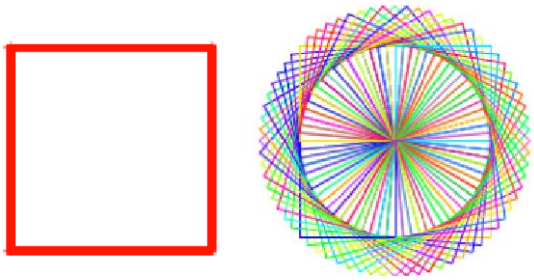
No *spriter* Caneta, coloque na área de *script* o bloco **[quando receber < >]** e passe como parâmetro o valor "*Polígono*".

O *script* que está abaixo do bloco **[quando clicar em ⚑]** (criando anteriormente) arraste do bloco **[levante a caneta]** para o bloco que acabou de colocar, deixando no bloco **[quando clicar em ⚑]** apenas o bloco **[apague tudo]**, conforme é apresentado na Figura 46.

Figura 46 – Organização do *script* do polígono

Agora vamos criar o *script* para atender o botão "Arranjo". Eu quero que o botão "Polígono" produza uma imagem regular e o botão "Arranjo" crie um efeito na tela com a sobreposição rotacional do polígono (Figura 47).

Figura 47 – Polígono e Arranjo executado na tela



Vamos começar duplicando o *script* do Polígono: basta clicar com o botão direito no topo e clicar em duplicar.

Figura 48 – Duplicação de *script*

Toda a parte que está pronta será aproveitada, todavia, para ficar um pouco diferente coloquei a caneta para 1, e inicie a cor também com valor 1.

Agora vamos acrescentar mais um *loop* **[repita ( ) vezes]**#e*stou passando o valor 72 vezes. Isso está relacionado com o valor em graus que acontecerá* #. O *loop* anterior vai ficar dentro desse novo *loop*.

Mas eu quero mudar a cor da caneta durante a execução; então acrescento o bloco **[mude a cor da caneta para ( )]**; nesse parâmetro eu vou passar o bloco de operadores **[número aleatório entre ( ) e ( )]***#estou sugerindo os valores "-500" e "500"*#, caso queira pode usar uma faixa menor. E abaixo do *loop* repita quatro vezes (ainda dentro do *loop* de 72 vezes): vou colocar para ir para a coordenada (0,0) e girar 5º graus (esses cincos graus está relacionado com o *loop* de 72 repetições), pois 5 * 72 = 360.

Figura 49 – *Script* do Arranjo

Análise do *script*

Para usar a caneta é simples: basta ter uma área de *script* com ator vazio, bem simples. Outro ponto interessante que vimos é que ao duplicar um *spriter,* é copiado tudo incluindo o *script* dele. Contudo, esse tipo de facilidade deve ter bastante atenção, pois devemos observar cada valor informado para não dar confusão, como a posição em que corrigimos a mensagem enviada ao clicar nos botões.

Observamos que a caneta é bem simples e depende exclusivamente de criatividade, como no *script* inicial (Polígono) para o segundo (Arranjo) apenas colocamos mais um *loop*, três blocos e mudou totalmente o efeito.

Também podemos controlar alguns erros com eventos simples, como o tamanho do polígono. Esse, resolvemos definindo os valores máximo e mínimo.

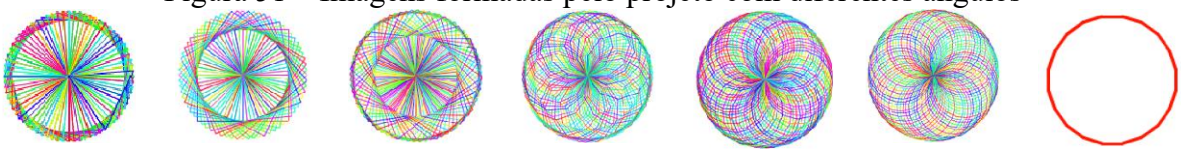
Ficou fácil, houve uma interação básica, onde o usuário mudou o botão deslizante e clicou no botão para criar o polígono. Porém, só podemos fazer polígonos de quatro lados? Bem se um polígono regular tem as arestas e ângulos iguais, podemos fazer outros polígonos. Existem duas formas para executar essa questão: a primeira é criando outra variável e usando o botão deslizante também. Aí devemos pensar: se o ângulo de 90º aconteceu 4 vezes, pois 4*90 = 360, logo o valor de 360º é dividido pelo número de ângulos, ou seja, para triângulos (360/3) e por aí...

Outra dica de interação: é bem interessante perguntar ao usuário "Construir um polígono regular de quantos lados, nesse caso, é só usar o bloco **[resposta]** para o número de repetições e o valor de 360/**[resposta]**. Esse é o desafio, criar uma interação para o usuário definir o tipo de polígono que deseja.

Figura 50 – *Script* e saída na tela para o uso de pergunta

No caso da interação por perguntas, devemos fazer teste. Imagine que o usuário digite 1 ou 2, seria possível criar esse polígono? A questão é o máximo de arestas. Observe a Figura abaixo em que coloquei de 3 a 20 ângulos. Com 20 ângulos, já é quase uma circunferência.

Figura 51 – Imagens formadas pelo projeto com diferentes ângulos



3 ângulos | 4 ângulos | 5 ângulos | 10 ângulos | 15 ângulos | 20 ângulos | 20 ângulos

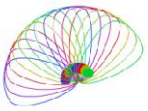
#fica@dica

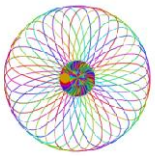
Haveria outro processo para criar efeitos com imagens prontas, como foi realizado com a caneta?

Digamos que queira fazer um arranjo com outra forma (ou outro procedimento qualquer) e não consigo desenhar com a caneta, por ter muitas instruções. Nesse caso, se não

consigo desenhar com os blocos da caneta e termos a forma pronta, podemos usar a forma como spriter e usar o bloco ***[carimbe]****. Assim teremos um efeito próximo, dentro do que foi desenvolvido pela caneta usando o carimbo. Ou, caso deseje, poderá utilizar esse bloco de várias outras formas.*

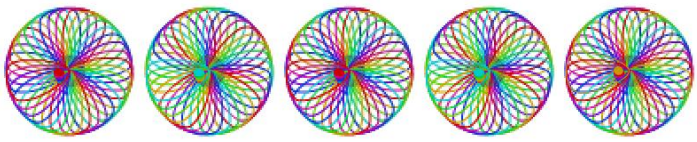
Forma primária (ator) ***Desenho no palco em andamento*** ***Desenho finalizado***





Vamos trabalhar como um spriter qualquer e definir uma limpeza de efeitos, limpar a cor e apontar a direção: ***[apague tudo]****;* ***[mude o efeito <cor> para (0)]****;* ***[aponte para a direção (90) graus].*** *Após isso criamos o loop para fazer as imagens, com o número de repetições para deslocar-se no palco, definir a posição (usamos variáveis, que inicializamos com "-180, 0") e ao fim incrementamos de 80 o valor de x, isso fará o deslocamento:* ***[vá para x:([x]) y:([y])]****;* ***[adicione a <x> (80)]****. Criamos outro loop para fazer a figura e definir que seja carimbado, girar 10º a imagem (observar ângulo pelas repetições até 360) e mudar o efeito:* ***[carimbe]****;* ***[gire (10) graus]****;* ***[adicione o efeito <cor> (25)]****.*

Script ***Apresentação na tela***



Vídeo deste projeto

Script do projeto

https://youtu.be/lEEOf6xE164

Script da etapa de interação

https://youtu.be/OW6HKFOjD5E

5º Tópico: Criação de menu

A finalidade dessa etapa é criar um menu de interação, mudar o pano de fundo e iniciar o projeto final.

4º projeto: menu inicial - efeitos, posicionamento e pano de fundo

Vamos desenvolver uma tela inicial para quando ocorrer mais de uma informação. Ou seja, quando houver a necessidade de desenvolvermos mais de um evento em telas que necessitem trocar o ambiente totalmente.

Botão menu

Vamos criar um botão na base do canto esquerdo da tela de quem observa (tipo do *Windows*). Dentro do botão, uma seta para baixo ou para cima ("ou" quer dizer para cada fantasia).

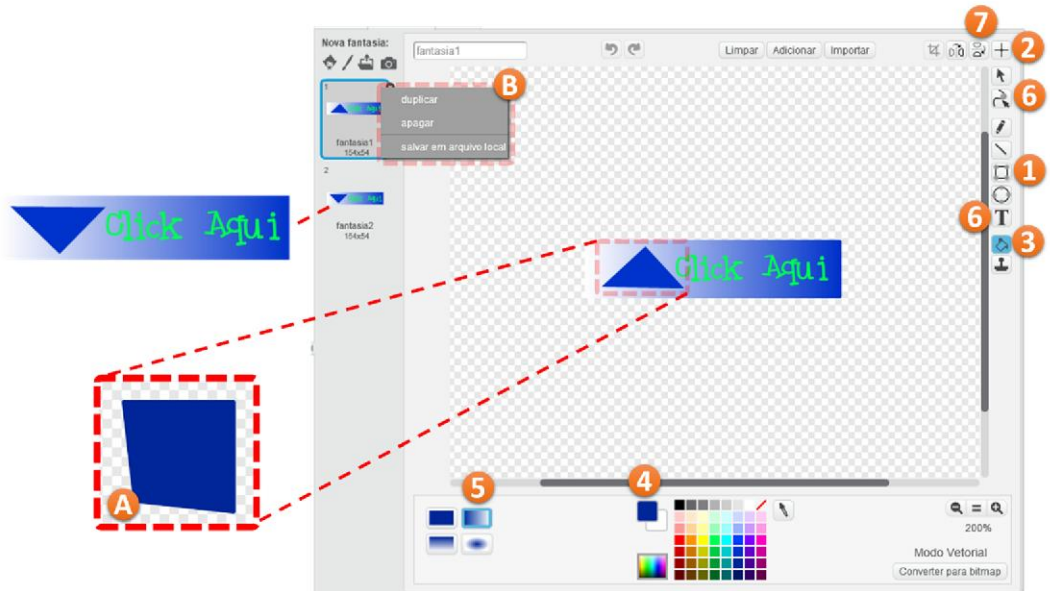
Para criar o botão do menu inicial, vamos utilizar a ferramenta "Pintar um novo ator" (pincel abaixo do palco ). Na área de criação, selecione o modo vetorial.

Observe a Figura 52 e siga os passos indicados pelas letras e números. Inicie criando um retângulo (1), centralize (2), pinte o retângulo (3). Eu sugiro um gradiente entre azul e branco; para isso selecione o azul e o branco (4) e a ferramenta de coloração gradiente (5), para girar use o espelhamento horizontal (7). Digite o texto "*Click* aqui".

Crie um triângulo. Para o triângulo, use o a mesma ferramenta do retângulo (1), após criar o quadrado, *clique* na ferramenta remodelar (6) e dê um duplo *click* em um dos vértices (A). Dessa forma desaparecerá o vértice clicado. Gire a figura, para que apareça um triângulo.

Observe que para criar um quadrado, pode aumentar o *zoom* e contar os quadradinhos do fundo, sendo igual para altura e largura.

Figura 52 – Criação do botão menu



Depois de criar, duplique a fantasia (B) e gire apenas o triângulo com a ferramenta de espelhamento vertical (7).

Nosso objetivo é que esse botão acione o painel e sejam apresentadas as opções do menu. Então vamos criar um *script* que faça o botão enviar mensagens para todo projeto.

Primeira dica: coloque o ator (botão) na posição que deseja – estamos sugerindo no canto inferior esquerdo de quem observa a tela – essa dica ajuda para que, quando levar o bloco da posição, ele já vá com as coordenadas posicionadas no bloco.

Crie uma variável (sugerimos o nome "ClkOn")#*aparecerá o bloco* ***[ClkOn]*** *mais quatro blocos de manipulação do bloco criado*#, que servirá para indicar quando esse botão foi clicado. Crie também uma lista (sugerimos o nome "seleção" #*surgirá o bloco* ***[Seleção]*** *mais nove blocos para manipular a lista*#).

Arraste o bloco de **[quando clicar em ⚑]**, para área de *script*, junte o bloco **[vá para x:( ) y:( )]**#*se posicionou o ator onde deseja, esse bloco vai preenchido com a posição correta que irá funcionar*#. Para garantir que ficará de frente junte o bloco **[vá para frente]**.

Arraste para área de *script* o bloco da lista **[apague (*1*) de <*Seleção*>]** e arraste o bloco **[mude <*ClkOn*> para (*0*)]**#*preencha com valor zero*#, junte-os na pilha. Junte o bloco da variável lista **[insira (*Start*) na posição 1 de <*Seleção*>]**. Essa etapa fará com que a cada vez que for iniciado, será apagado o valor que estiver na variável e na lista, sendo "setados" os valores de zero e "*Start*".

Coloque o bloco para garantir que o ator inicie com a fantasia 2 do botão.

Para criar um "encanto", vamos dar efeito de mudança de cor, como se estivesse piscando. Então vai acontecer sempre, para isso coloque um bloco **[sempre]**, dentro do *loop* sempre; vamos garantir que isso aconteça até que seja selecionado uma opção, para isso use um bloco **[repita ( ) vezes]**. Para preencher essa condição, vamos usar o bloco de operação = e comparar se o valor será igual ao que iniciamos "*Start*" **[([*Seleção*]) = (*Start*)]**.

Dentro do *loop* do bloco **[repita ( ) vezes]**, coloque o bloco **[adicione o efeito <*cor*> (*5*)]**#*observe que sugerimos que insira o valor "5" e adicione o efeito* #.

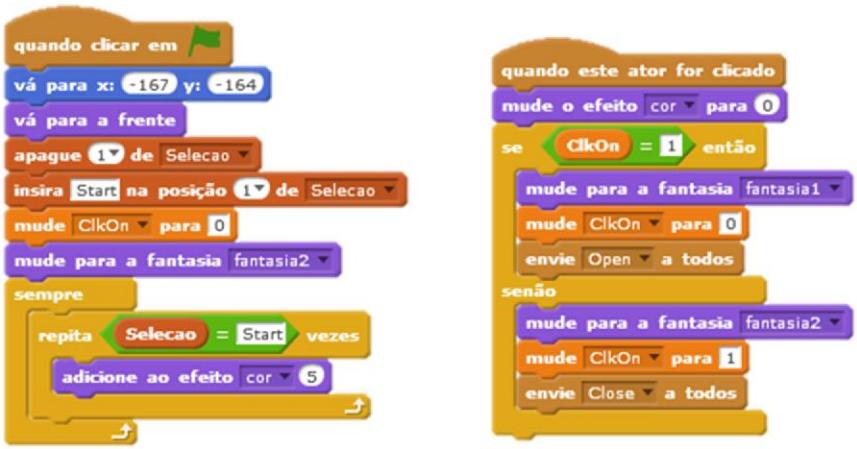
Após essas etapas, criamos o evento botão para quando for iniciado. Agora vamos fazer para quando ele for clicado, será quando abre o pinel ou esconde.

Arraste o bloco **[quando este ator for clicado]**, abaixo dele vamos para o efeito de mudar cor, pois o usuário já entendeu que esse botão apresenta o menu. Coloque o bloco **[mude o efeito <*cor*> para (*0*)]**. Note que anteriormente foi utilizado o bloco com valor adicione, ou seja, sempre será dado mais o valor (5 + 5 + 5 ...), agora usamos o bloco com valor mude e demos o valor "0", ou seja, limpamos todos os valores, seja qual for que esteja selecionado.

A seguir façamos um teste: use o bloco **[se < > então, senão]**, preencha com o bloco operador, realizando uma comparação entre o valor da variável "*ClkOn*" e o número "1"**[(*[ClkOn]*) = (*1*)]**. Caso a sentença seja verdade, entrará na opção "se, então" do primeiro *loop*. Nesse primeiro *loop,* coloque o bloco para alterar a fantasia **[mude a fantasia <*fantasia1*>]**, mude o valor da variável para não entrar novamente nesse *loop* **[mude <*ClkOn*> para (*0*)]** e envie a mensagem em que foi aberto o painel como bloco **[envie <*Open*>a todos]**#*você cria a mensagem*#. No segundo *loop* ("*senão*") copie esses blocos do 1º *loop* e mude o valor para "fantasia 2", "ClkOn para 1" e a mensagem será "*Close*". Assim, caso seja clicado o botão envia a mensagem e ao mudar o valor da variável "*ClkOn*", ele mesmo controla a posição do botão.

Note o *script*: quando é clicado, muda a fantasia, torna o valor "0" e envia uma mensagem que está aberta, assim quando ele for clicado novamente o valor é zero, então ele não entra nesse *loop*, mas no *loop* "senão". No *loop* "senão" ocorre o contrário.

Figura 53 – *Script* do botão menu



O que é lista?

O bloco lista tem a mesma função do bloco variável, porém na lista é possível armazenar mais informação que apenas uma variável. As listas são úteis para armazenar um conjunto de valores.

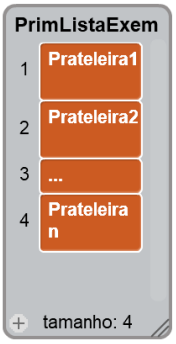
Vamos pensar no bloco variável como uma caixa que contém uma informação, ou seja, um valor, já a lista vamos pensar em uma estante com muitas prateleiras e cada prateleira recebe uma informação (caixa). Como pensamos em prateleiras, é necessário entender que cada valor tem um índice, ou seja, cada prateleira.

Para nomear uma lista, deve fazer o mesmo procedimento da variável: 1° caractere maiúsculo e demais minúsculos, em caso de nome composto segue essa ideia sem espaço e outros tipos de caracteres especiais.

Assim como no bloco variável, ao criar uma lista é possível armazenar qualquer tipo de informação e ao criar uma lista são criados os blocos de instrução da lista.

Blocos da lista

Lista

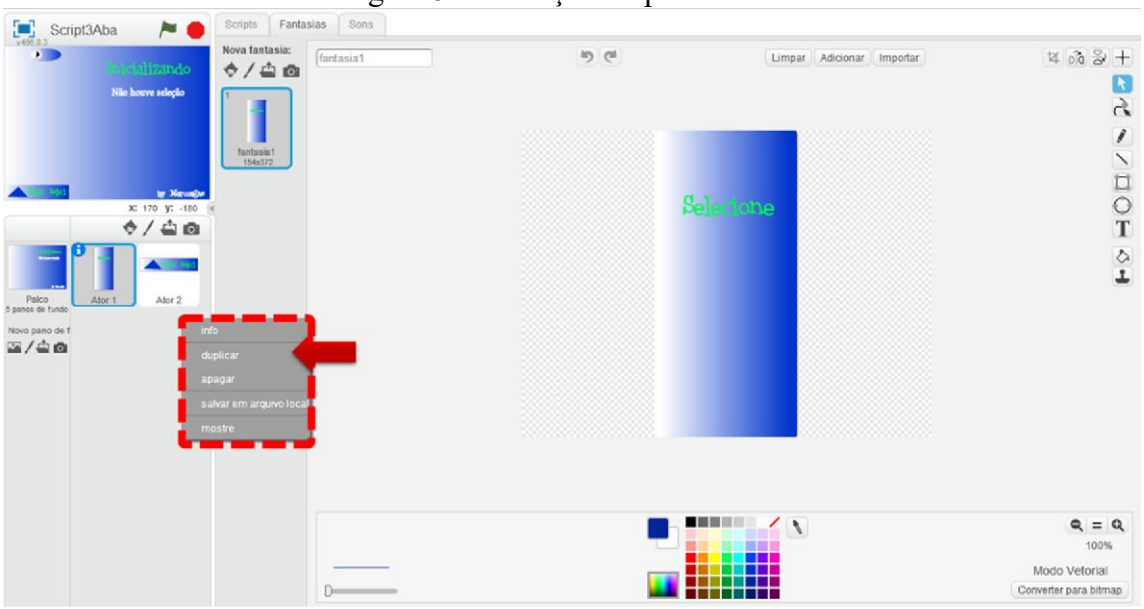


Painel menu

Após o botão, vamos criar a barra que irá se elevar, quando o botão for clicado. Nesse caso, ao invés de criar o retângulo, vamos aproveitar o botão que já definiu o tamanho, então duplique o *spriter* do ator do botão (**atenção** na etapa anterior duplicamos a fantasia, agora vamos duplicar o *spriter,* será fora da edição - *Paint*. Use a área de *spriters* para duplicar esse *spriter*) - destaque na Figura 54.

Após duplicado, vamos editar o segundo ator. Na aba de fantasias, usando o *paint*, apague a segunda fantasia. Na primeira, apague o triângulo e mude o texto para selecione (isso servirá para alertar ao usuário que deve selecionar o que está na barra). Aumente o retângulo na **altura**, para que da base, vá até a altura. Tome cuidado para **não alterar** a **largura**. O objetivo é que esse ator apareça abaixo do botão anterior.

Figura 54 – Criação do painel menu



Agora vamos criar o *script* do painel. Para o painel, ele vai obedecer aos eventos definidos no botão. Para garantir que não apareça no palco, arraste o bloco de **[quando clicar em ⚑]**, para área de *script*, coloque abaixo o bloco **[esconda]**.

Arraste o bloco **[quando receber <*Open*>]** e abaixo dele coloque o bloco **[mostre]**; isso vai garantir que o painel apareça, pois, no evento anterior mandamos que ele se ocultasse. Após isso, coloque o bloco para a posição (lembra da dica do botão? Faça o mesmo: posicione antes de usar o bloco). Dessa vez vamos usar um bloco para que o ator vá para sua posição lentamente, use o bloco **[deslize por (*1*) seg até x: ( ) y: ( )]***#já vem preenchido com a posição#*. Abaixo vamos garantir que o painel fique atrás dos outros atores, por isso coloque o bloco **[vá (*2*) camadas para trás]**.

Agora vamos fazer com que aconteça o evento, quando for clicado o botão para "Close". Arraste o bloco **[quando receber <*Close*>]** e abaixo dele coloque o bloco **[deslize por (*1*) seg até x: ( ) y: ( )]***#já virá preenchido#*; altere o valor de y para -400 e caso não dê, pode tentar outros valores e por último (observe que a ordem foi trocada intencionalmente, agora que colocaremos o bloco para omitir o ator) coloque o bloco **[esconda]**.

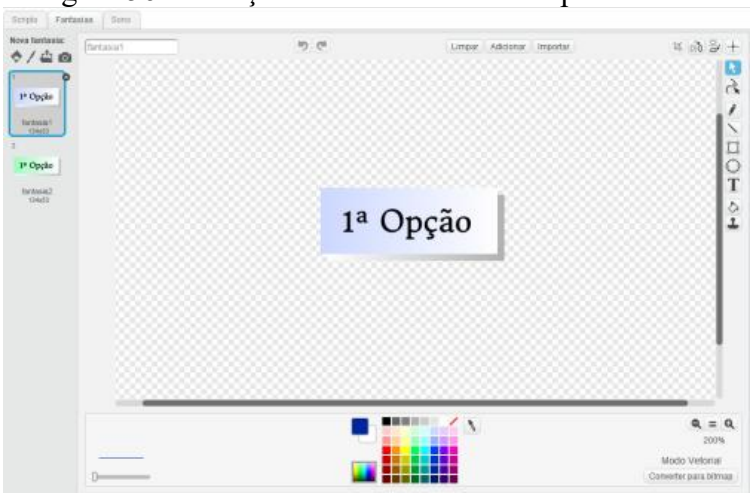
Figura 55 – *Script* do painel menu

Opções de menu no painel

Vamos criar o botão para carregar o valor. Sugiro um retângulo menor que o anterior. Proceda da mesma forma que criou o botão. Caso queira dar um efeito de profundidade, crie um retângulo, preencha-o com o gradiente cinza e branco e o deixe embaixo. Crie outro retângulo e o posicione por cima do anterior. Sugiro uma coloração diferente do retângulo maior (azul claro e branco). Coloque um texto qualquer que indique esse botão.

A seguir duplique a fantasia e mude a cor da segunda fantasia. Sugerimos um tom de verde claro (essa mudança de cor vai indicar que esse botão foi selecionado).

Figura 56 – Criação do botão de eventos para o menu



Da mesma forma que a duplicação de *spriter* (Figura 54), duplique esse *spriter* e edite para alterar o texto. Sugerimos textos: "1ª Opção" e "2ª Opção".

Para o *script* do botão, vamos arrastar o bloco de **[quando clicar em ⚑]**, para área de *script*, coloque abaixo o bloco **[esconda]**. Em seguida, vamos fazer dois testes sempre, então use o bloco **[sempre]** e dentro do *loop* sempre faça um teste com bloco **[se < >, então]**, preencha o valor de teste com o bloco operação **[não ( )]**, dentro desse bloco, passe como parâmetro o bloco de comparação entre o valor da lista seleção com 1ª Opção, use o bloco **[(*[Selecao]*) = (*1Opção*)]**. Esse teste verifica se não foi selecionado a 1ª Opção. Dentro desse *loop,* coloque o bloco **[mude para a fantasia <*fantasia1*>]**.

Façamos outro teste. Ainda dentro do bloco **[sempre]**, copie o bloco inteiro do teste "se, então", com todos os blocos e altere: retire apenas o bloco **[não ( )]**, mas tenha o cuidado de manter o teste da seleção. No bloco da fantasia mude para 2. Ou seja, o que estamos fazendo é: se não for a 1ª Opção, mantenha a fantasia 1, caso seja clicado mude para fantasia 2.

Arrasta agora o bloco **[quando este ator for clicado]**, abaixo coloque a mudança do valor na lista para 1ª Opção#*note que eu não usei o cedilha ou acentuação gráfica "~" em nenhuma opção e, que também usei o bloco substitua, não o bloco insira*#.

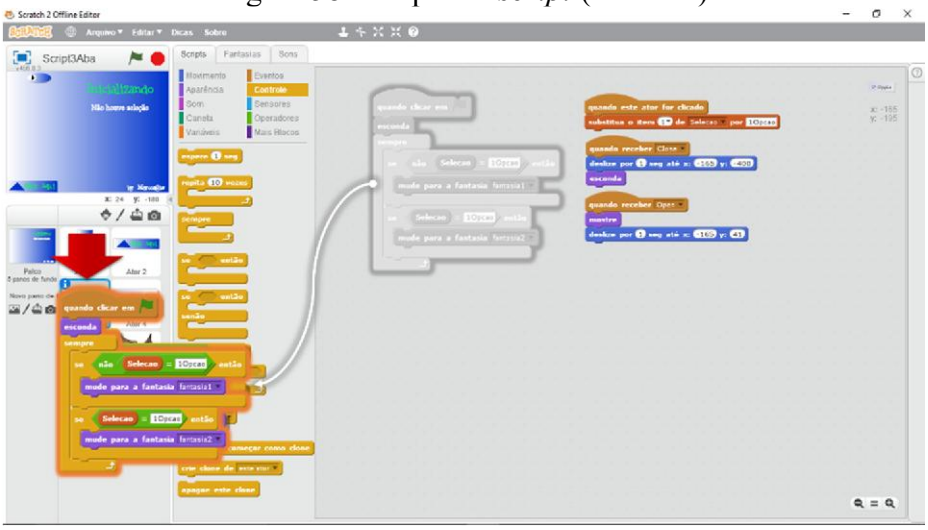
Arraste o bloco **[quando receber <*Open*>]** e abaixo dele coloque o bloco **[mostre]**, já comentamos esse formato no painel. Após isso, coloque o bloco para a posição (lembra da dica do botão). Novamente vamos usar um bloco **[deslize por (*1*) seg até x: ( ) y: ( )]***#já vem preenchido com a posição#*.

Agora vamos fazer para o evento "Close". Arraste o bloco **[quando receber <*Close*>]** e abaixo dele coloque o bloco **[deslize por (*1*) seg até x: ( ) y: ( )]***#já virá preenchido#* altere o valor de y para -400 e também coloque o bloco **[esconda]**.

Figura 57 – Criação do 1 *script* dos botões de menu

Para criar o *script* do segundo botão, vamos simplesmente arrastar para o *script* total para o outro *spriter* (esse movimento corresponde ao evento "mochila" da versão *off-line*).

Figura 58 – Cópia do *script* (Mochila)



Observe na Figura 58, que ao levar de um *spriter* para o outro, o que irá receber ficará azul (seta vermelha em destaque na figura). Após fazer isso com os dois *scripts*, vai observar que no outro *spriter* tem um *script* **igual** e nesse também. Logo, se é igual, irá executar as mesmas coisas, na mesma sequência e posição do outro. Mas não é isso que queremos, por isso, o segundo botão deverá acertar os valores de posição e Opção para "2Opcao".

Para o nosso arquivo, sugiro o valor de "-35" para a ordenada y: no bloco do "open".

Figura 59 – Criação do 2 *script* dos botões de menu

Já é possível cumprir a proposta. Agora vamos criar alguns melhoraremos nesse arquivo, para que fique mais inteligível nossa proposta.

Pano de fundo

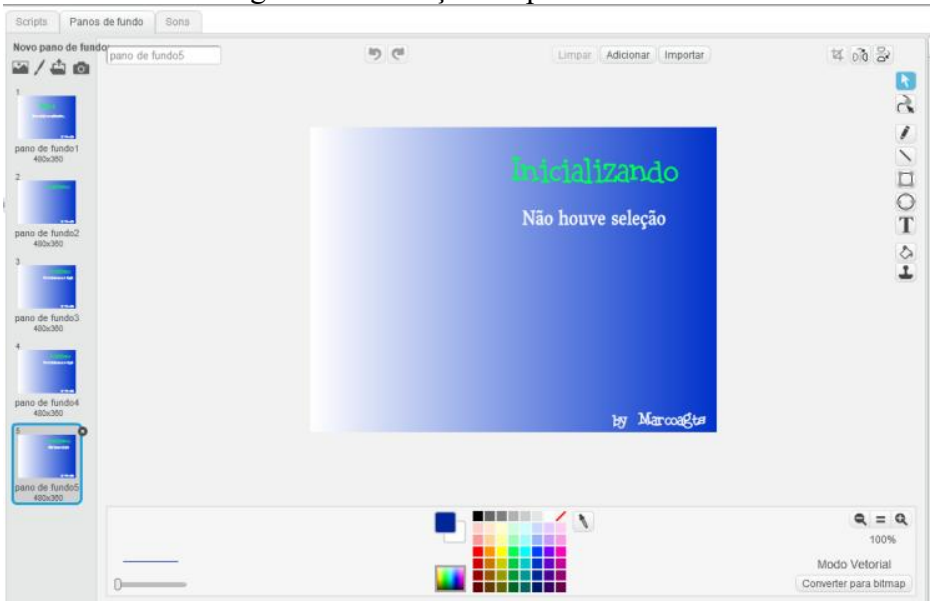
Vamos criar um pano de fundo que deixe clara a posição que está acontecendo na tela.

Vá em palco, no canto esquerdo da tela, e na aba "panos de fundo", mude para vetorial e crie um quadrado do tamanho da área total. Mude a cor (estamos utilizando um gradiente em azul e branco) e coloque o texto "Inicializando" e abaixo "Bem-vindo ao aplicativo…", caso queira, coloque seu nome abaixo.

Também duplique esse pano de fundo e edite o texto para "Inicializando".

Faça esse processo por cinco vezes e mude o valor do texto para "Inicializado" e "1ª Opção", depois o mesmo com "2ª Opção" e por último "não houve seleção. Esses textos serão mostrados conforme for selecionado o pano de fundo.

Figura 60 – Criação do pano de fundo



No *script,* podemos para o primeiro, ao iniciar, colocar o pano de fundo 1(mensagem: Seja bem-vindo…), quando abrir o pano de fundo 2 (mensagem: Inicializando). Depois disso é necessário fazer teste para saber se foi selecionado e qual foi a seleção. Ao fechar o painel (mensagem Close) deve ser testado se o valor que está na lista é "*Start*" (valor inicializado) e nesse caso apresenta a mensagem do pano de fundo 5 ("Inicializando", "Não houve seleção").

Se foi clicado o 1º botão a mensagem é "1Opcao", então muda o pano de fundo para 3 e aparece a mensagem: "Inicializado", "Você selecionou a 1ª Opção". O mesmo para o 2º botão, porém como pano de fundo 4 e a mensagem: "Inicializado", "Você selecionou a 2ª Opção".

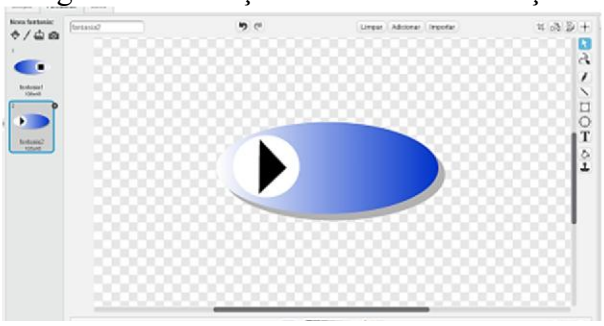
Figura 61 – *Script* do pano de fundo

Botão indicativo de que está rodando

Vamos criar um monitor para nosso arquivo. Esse monitor servirá apenas para dizer que o sistema está *on* ou *off-line*, ou seja, funcionará junto com o pano de fundo.

Sugiro uma elipse, como fizemos com o retângulo e os botões. Colocamos uma por baixo, na cor cinza, e dentro da elipse, na parte clara, colocamos uma circunferência e dentro, um quadrado que simboliza "*stop*". Depois duplica, e o quadrado passa a ser um triângulo. Tudo como fizemos anteriormente.

Figura 62 – Criação do botão de indicação



Para fazer o *script*, vamos colocar na posição fixa, vamos dar efeito de fantasma quando ele estiver na execução do aplicativo.

Após criar, precisei alterar o tamanho para 70% do original; caso precisem, coloquem o bloco **[mude o tamanho para ( ) %]***#pode usar para diminuir ou aumentar#*. Coloque na posição que deseja e arraste o bloco; vá para. Defina a fantasia 1.

Quando iniciar a barra de menu, vamos colocar na posição da fantasia 1. Vamos colocar um efeito de fantasmas ao contrário, coloque o bloco para repetir 15 vezes e dentro o efeito fantasmas "-5". Esse efeito o fará aparecer e quando fechar, ele irá esmaecer.

Assim, o bloco de aba de menu fechada (Close) repete esse mesmo, porém, com fantasia 2 e com o efeito fantasma "5". Esse processo o fará esmaecer e não atrapalhar a interação.

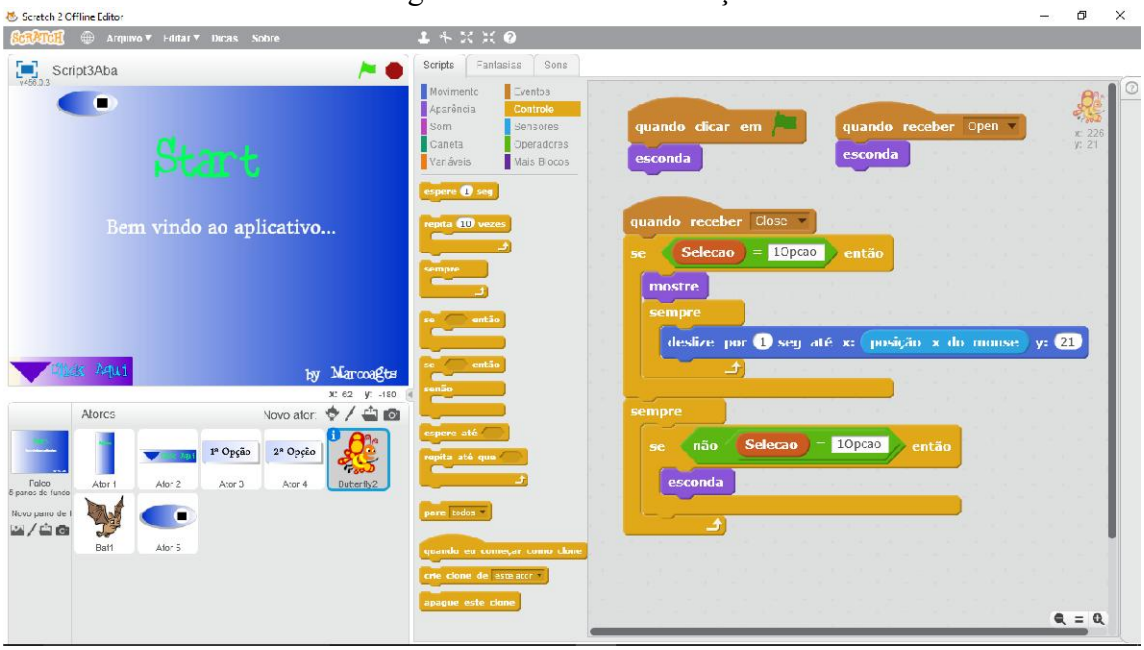
Figura 63 – *Script* do botão de indicação

Já ficou interessante, mas vamos criar um movimento simples. Servirá apenas para podermos observar o efeito do botão mudando os atores.

Colocamos no palco um *spriter* pronto do programa, uma borboleta amarela. O *script* dela é simples. Esconder ao iniciar o programa e quando o painel for acionado (mensagem *Open*). Quando for enviada a mensagem *Close,* devemos fazer um teste: primeiro, ver se a "seleção" contém "1Opcao"; se for positivo, executo o *loop* mostre, já que estava escondido e o bloco sempre com o bloco para deslizar até a posição "x:" do mouse. Usei o bloco de sensor **[posição x do mouse]**.

Depois outro teste com um bloco sempre e um se não for a 1ª opção, para esconder. Esse usamos o bloco **[sempre]**, dentro do *loop* o bloco **[Se < > então]**, na condição o Operador **[não ( )]** e passei como parâmetro o teste se o valor de "**[Selecao]**" = "1Opcao". Ou seja, se não for a 1ª Opção se esconda sempre.

Figura 64 – Tela da animação



Inserir outro *spriter*, dessa vez um morcego e copiei (mochila) os *scripts* do objeto anterior. Fiz a alteração apenas na opção, para "2Opção".

Analisando o 4º projeto da barra de menu

Nesse arquivo, vimos o efeito fantasma, a mochila, o pano de fundo, passamos mensagem e recebemos mensagens. Aprendemos a controlar as mensagens. Também vimos que ao duplicar um *spriter* ou fantasia toda informação é copiada.

Demos efeitos simples da sombra e efeitos empolgantes, como o fantasma e a troca de cores. Existem outros que podem ser bem interessantes:

Nota (todos os efeitos acima são gerados da imagem original, sem acumular efeitos): (1) limpa efeitos gráficos; (2) muda cor; (3) efeito olho de peixe no centro da imagem; (4) repete a imagem sem alterar o tamanho; (5) cria efeito de baixo número de pixel; (6) deforma o centro da imagem com efeito de rodamoinho; (7) diminui o brilho (embranquece); (8) cria uma transparência; (9) demonstra a possibilidade de inverter o efeito, no exemplo o de brilho que escureceu a imagem.

Passamos blocos parâmetros como um parâmetro, para outros parâmetros (negamos uma afirmação) e aprendemos que o bloco **[repita ( ) vezes]** pode ser uma condição, não só o valor.

Bem, agora vamos para o desafio dessa etapa: criar uma forma de limpar a informação da lista do bloco que criamos **[Selecao]**. Na verdade, ao limpar, estarei reiniciando, ou seja, estarei "resetando" o valor. Então, o que deve ser criado é um botão "reset". E o que esse botão fará? Ora, ele vai colocar o valor inicial na lista **[Selecao]**. Que valor é esse? Eu disse no texto, vamos dar uma relida para ajudar a fixar.

Vídeo deste projeto

https://youtu.be/qJxBgFCghI4

6º Tópico: (Re)Usabilidade e bloco clone

A finalidade dessa etapa é "remixar" projetos prontos, trabalhar com blocos algébricos e lógicos e identificar a criação e controle de clone.

5º projeto: aproveitamento, adaptação de arquivo pronto e clone

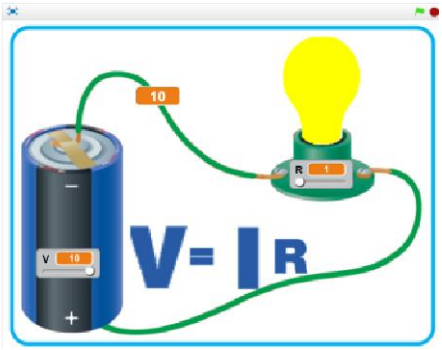
Vamos trabalhar um arquivo pronto na *internet* e fazer algumas alterações, isso é "*remixar",* na linguagem do scratcher. Se quiser buscar o arquivo na *internet* (repositório do scratch.mil.edu) vai achar algumas variações desse arquivo, que é denominado de "OhmsLaw", "Ohms" ou "Ohm'sLaw". O arquivo que vamos remixar é sobre a Lei de Ohm (V=RI). Esse assunto é comumente discutido na física, mas algumas vezes também é visto na química, quando é demonstrada a pilha de limão.

Ao trabalhar esse arquivo vamos incluir alguns assuntos e demonstrar como adaptar algo pronto.

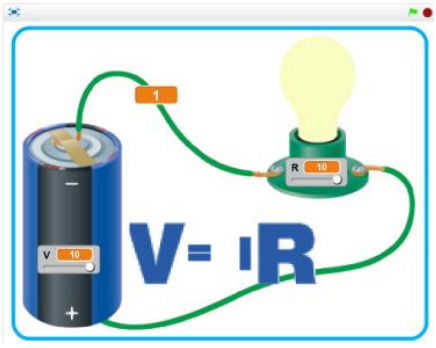
Inicialmente vamos entender o arquivo: a tela do arquivo apresenta uma pilha química, dois botões deslizantes (controle da tensão "V" e resistência "R"). A corrente aparece como resultante das variações entre esses dois botões deslizantes. Ao aumentar o valor da tensão, com a resistência baixa, aumenta a luminosidade da lâmpada e ao diminuir tensão ou aumentar a resistência diminui a luminosidade da lâmpada. Assim, o arquivo demonstra a relação entre a resistência, a tensão e a corrente. Observe a Figura 65 em relação à mudança dos valores de V e R.

Figura 65 – Controlando os parâmetros dos blocos variáveis

65A – Tensão elevada e baixa resistência



65B – Tensão elevada e igual resistência



Vamos entender como foi o desenvolvimento. A pilha, os fios, o bocal e a lâmpada apagada fazem parte do palco (Figura 66A), ou seja, é o cenário. A lâmpada (Figura 66B) é sobreposta no cenário, na mesma posição da outra e os caracteres (Figura 66C) que representam a equação de Ohm, se alteram conforme o valor no botão deslizante. Esses *spriters*, que representam a equação, são imagens independentes ("**V**", "**=**", "**I**" e "**R**").

Figura 66 – Imagens empregadas

Figura 66A - Palco

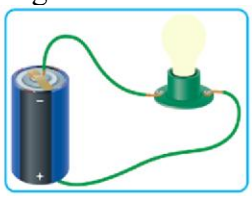
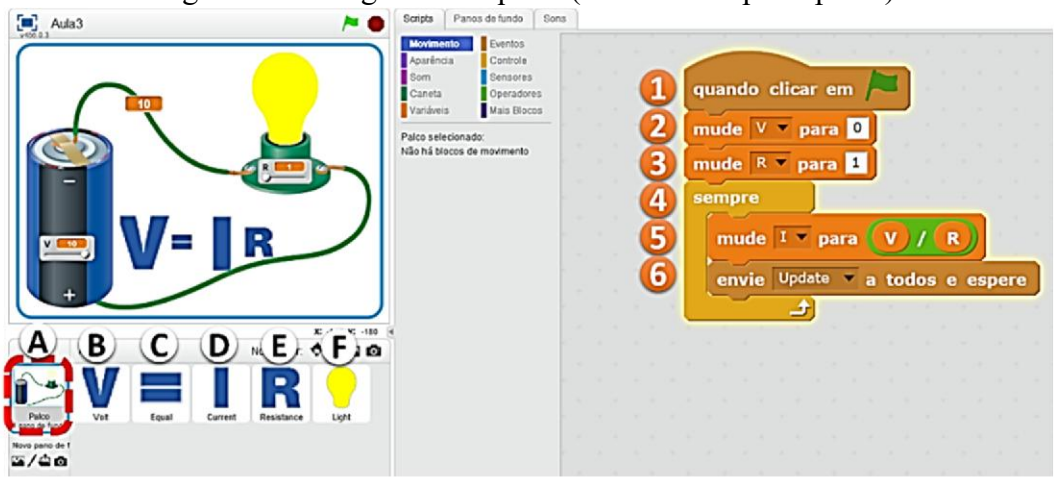


Figura 66B – Lâmpada

Figura 66C – Letras

Observe a próxima imagem (Figura 67), no banco de *spriters* estão todas as imagens que serão utilizadas: palco (A), caracteres "V", "=", "I" e "R" (B, C, D e E) e a lâmpada (F). No algoritmo existente do palco, o bloco **[quando clicar em ⚑]** (1) controla o início do arquivo. Os blocos **[mude < > para ( )]** ao lado dos círculos (2), (3) e (5) são a inicialização das variáveis para garantir o valor. No número (5), o valor está relacionado com a equação. O círculo do número (4) é o bloco para fazer em ato contínuo (*loop*) e o bloco (6) é para enviar uma mensagem <*update*>.

Figura 67 – Imagem do arquivo (visão do script do palco)



Se observar o número (3) da Figura 67, referente ao bloco variável **[mude <*R*> para (*1*)]**, é iniciado com valor 1. Isso é para garantir que não seja apresentado o valor "NaN", um termo em inglês "*Not a Number*". Esse valor "NaN" ocorre, porque na área de computação deve trazer sempre alguma informação e no caso de divisões com resultados de valor não definido, ou não representável, apresenta o "NaN". Por isso, o autor definiu que esse valor sempre ter o valor mínimo de "1".

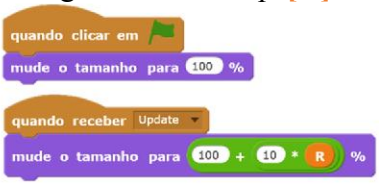
Os *scripts* que controlam os atores da equação no palco são simples. Para as três letras que representam a equação da Lei de Ohm, quando inicia o *script*, são apresentadas no tamanho de 100% do original, ao receber a mensagem *Update* (que está no palco) atualiza o tamanho para 100% do original mais (adição) do valor alterado pelo botão deslizante.

Figura 68 – Imagem do arquivo (visão do script das Letras)

Figura 68A – *Script*[V]

Figura 68B – *Script*[I]

Figura 68C – *Script*[R]



Para representar a luminosidade da lâmpada relacionada à corrente, a lógica é a mesma. O bloco utilizado é o efeito fantasma. A relação do efeito é dada pelo valor da corrente: com bloco **[mude o efeito <fantasma> para [(*100*) – ((*10*)* ([*I*]))]]**, altera a visualização da imagem no palco pelo valor da corrente.

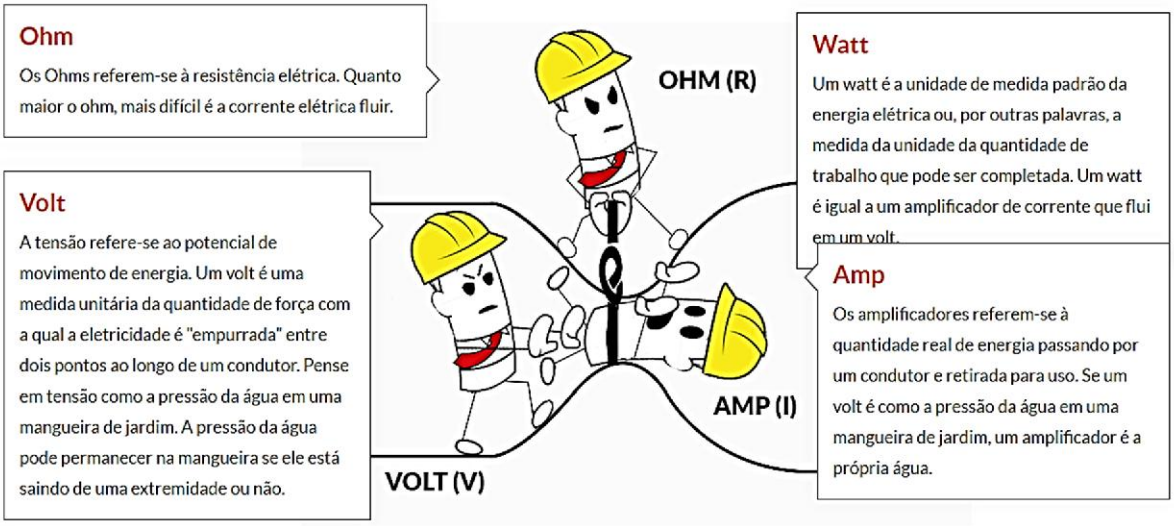
Figura 69 – Imagem do arquivo (visão do script do palco)

Assim, esse algoritmo simples termina empregando blocos de aparência e operadores, e os controles deslizantes para controlar o valor da variável e uma interação.

Agora observe que quando afirmamos da relação entre a corrente e a luminosidade estamos falando de um contexto abstrato. Quando explicamos para o aprendiz qual é o efeito da resistência no condutor, mencionamos em elétrons se deslocando pelo condutor.

Isso **não** fica claro visual no algoritmo que analisamos. Geralmente para essa etapa, desenhamos um condutor e colocamos bolinhas deslocando-se (indicado por vetor) em um condutor. Outra imagem muito interessante (engraçada) é a Tensão empurrando a Corrente e a Resistência impedindo o caminho (Figura 70)

Figura 70 – Relação da equação de Ohm (V=RI)



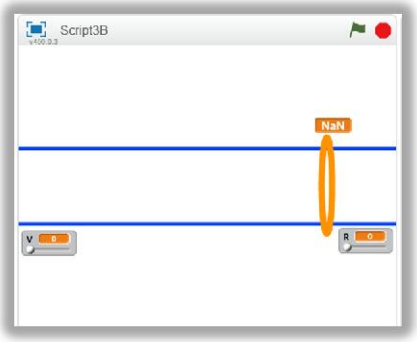
Fonte: Humphreys (2017)[2]

Vamos tentar reproduzir a Figura 70, porém um pouco mais próxima do que comentamos em sala.

O objetivo é desenhar um condutor elétrico com "bolinhas", deslocando de um lado para o outro, e que possamos deformar esse condutor, quando aumentamos a "resistência".

[2] HUMPHREYS, Evan. **O que é uma bateria solar?.** 2017. Disponível em: <https://goo.gl/CBHiJM>. Acesso em: 28 jul. 2017.

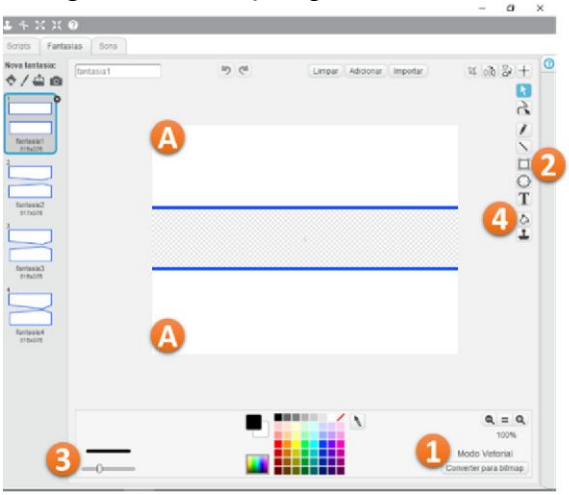
Figura 71 – Tela condutor (corrente elétrica)



Então, para começar a alteração no *script,* é necessário fazer um planejamento. Minha sugestão é que tenhamos uma tela inicial com a possibilidade de escolher qual o evento que queremos (Exatamente o projeto anterior).

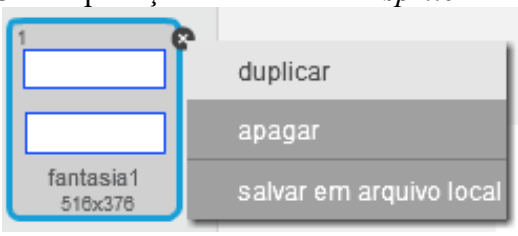
Para criar o *spriter* é simples. Vamos observar a Figura 72: selecione o pincel para criar uma *spriterr*, no ambiente de criação selecione modo vetorial (1), crie dois retângulos (2): um na parte superior e ou outro na parte inferior (A) (o objetivo de fazer dois retângulos é para esconder as bolinhas que podem sair do caminho). Selecione uma linha média (3) e de cor branca. Amplie o retângulo para que fique maior que o ambiente. Selecione o balde de tinta (4) e uma cor (sugestão o azul). Passe suavemente pela borda, assim que passar na posição correta vai ficar colorido, é só dar um *click*.

Figura 72 – Criação *spriter* do condutor



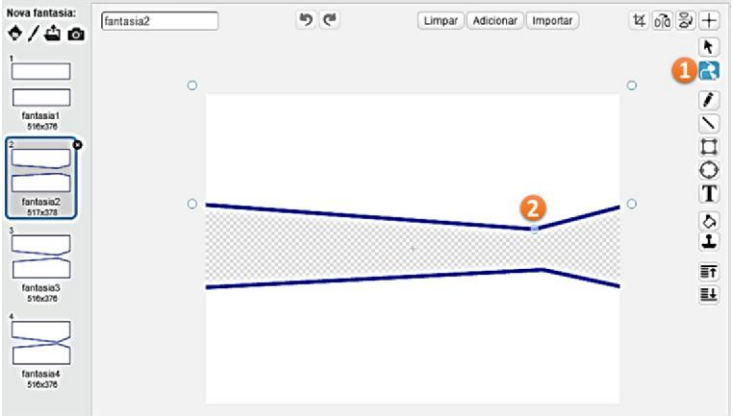
Com botão direito na fantasia,*click* em duplicar.

Figura 73 – Duplicação da fantasia do *spriter* do condutor



Para a segunda fantasia, observe a Figura 74: use o botão remodelar (1) e com um *click* sobre a linha do retângulo mude a posição da linha (2). Faça o mesmo em baixo. Repita esses passos: duplicar e remodelar mais duas vezes. Dessa forma teremos 4 fantasias.

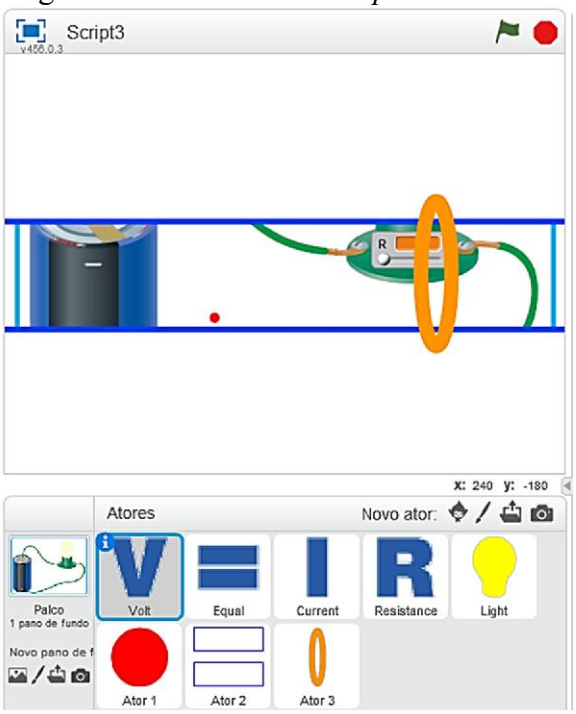
Figura 74 – Alteração do detalhe do *spriter* do condutor



Para o segundo *spriter* é só criar uma pequena bolinha no centro da tela. O terceiro *spriter* é apenas uma elipse estreita. Nenhum desses dois *spriters* terá fantasias no palco, apenas o condutor.

Ao final, nossa tela está confusa. Não é visualizada a imagem que começamos, nem conseguimos entender o que está sendo desenvolvido.

Figura 75 – Tela final com *spriters* criados



Scripts

Em cada um dos novos elementos (Ator 1, Ator 2 e Ator 3) vamos colocar o bloco **[quando clicar em ⚑]** e abaixo o bloco **[esconda]**.

De forma rápida, arrumamos nosso palco. Agora vamos criar o *script* da bolinha que representará a corrente elétrica.

Inicialmente vamos organizar. Lembre-se do que comentei, ao afirmar que é necessário fazer um planejamento: uma tela inicial que chama os eventos que queremos? Então, esse

planejamento já foi parcialmente inicializado quando desenvolvemos no arquivo do Projeto 4, o menu em aba.

Agora, iremos dar um nome ao evento, sugiro "CorrenteEletrica" e para o arquivo que usamos pronto, vamos nomear de "EquacaoOhm". Quando o evento sobre a corrente elétrica for acionado (1ª Opção), o ambiente de Ohm deverá ficar oculto e quando o evento de Ohm for acionado (2ª Opção), os atores que representam a corrente elétrica ficarão ocultos.

Para o Ator bolinha (corrente), já temos o *script* que, quando o aplicativo iniciar, ela deve ficar oculta.

Mas, o que é a corrente elétrica quando representamos? Na verdade, são vários atores, com ações semelhantes, dentro de uma coerência, ou seja, os elétrons são simbolizados pelo deslocamento da bolinha no mesmo sentido. Isso nos permite utilizar um recurso denominado **clone**.

Utilizando esse ator, vamos criar cópias com movimento. Para isso, utilizaremos o bloco **[crie um clone de <esse ator>]** (na aba Controle). É importante saber que o clone tem o exato comportamento de quando foi criado. Então, a seguir vamos definir a posição do ator: a posição de "x:" (movimentos horizontais) será sempre a mesma, pois necessitamos a mudança em relação a "y:", mas devemos deslocar o ator dentro do "condutor", ou seja, devemos deslocar em "y:" da altura até a base (movimentos verticais: ↓↑). Para o *spriter* marque a distância de "y:", o que fiz ficou entre 20 e -40. Então, determino a criação do clone desse ator e o deslocamento novamente. Ou seja, esse movimento está dentro de um *loop* faça isso continuamente **[sempre]**. Para esse posicionamento, usaremos um bloco de valores aleatório com o bloco de movimento **[vá para x: ( ) y: (*[número aleatório entre ( ) e ( )]*)]***#vamos passar para o valor de x: -235 e para y: a variação de "20" e "-40". Esse valor é para que fique dentro do espaço "condutor", caso o seu não esteja dentro desse parâmetro, deve rever#.*

Detalhe importante: relembrando o desenho, se não houver tensão, não devemos ter o deslocamento dos elétrons. Então, vamos fazer um teste com o bloco **[se ( ) então]**, passar como parâmetro o bloco lógico, em que a tensão (bloco variável **[V]**) é verificada se é maior que zero. Mas pode ser que a tensão não seja a máxima. Para evitar um teste para cada valor de tensão, podemos generalizar matematicamente com bloco de espera recebendo como parâmetro o operador aritmético de divisão. Se o bloco **[espere ( ) seg]** responde em tempo de segundo, usando o valor "segundo" dividido pelo valor da variável tensão ("V"), teremos uma alteração de tempo. Essa relação matemática implica que quanto maior o valor de "V" (cociente na operação), menor o tempo para criar outro clone (se o tempo é pequeno, crio mais clones). Assim, quanto maior a tensão, representada por "V", maior a corrente. Parece confuso, mas na verdade é apenas o uso do bloco temporizador com valor referenciado a tensão **[espere ([(*0.5*)/([(*[V]*)]) seg]**, antes de criar o clone:

Para os clones, tem um bloco especial que inicia: **[quando eu começar como clone]**, abaixo desse bloco iremos montar o *script* de comportamento. Inicialmente vamos garantir que os erros fiquem ocultos; use o bloco **[vá ( ) camadas para trás]**, passe como parâmetros "3", adicione o bloco **[mostre].** O bloco **[mostre]** é necessário, pois o ator original está oculto, logo suas cópias ficam ocultas.

Depois devemos definir o movimento contínuo com o *loop* do bloco **[sempre]** e dentro desse bloco, fazer o clone se deslocar, no sentido 90º (para direita), com bloco **[mova ( ) passos]***#sugiro o valor 5#* e **[aponte para a direção (*90*) graus]**.

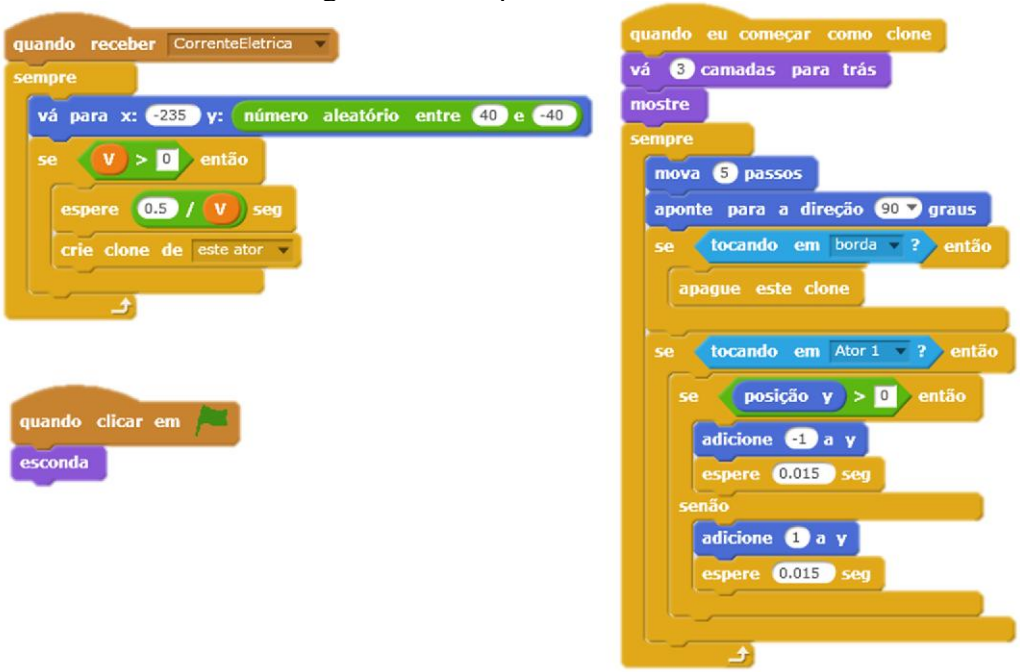
Agora vamos garantir o fim do clone, que para esse *script* será na borda da tela, então façamos o teste para que, quando tocar na borda, o clone seja apagado com os blocos: **[se ( ) então]**, passando como parâmetro o bloco **[tocando em <*borda*>]** e dentro desse *loop* o bloco **[apague este clone]**.

Então nos resta uma questão: quando acontecer a mudança da fantasia (alterar o valor da resistência), alterar a posição do clone para o interior, causando uma maior concentração de elétrons (lembra do aquecimento?); podemos demonstrar isso de forma animada. Então faremos um teste para verificar se o Ator 2 (elétrons) tocou no Ator 1 (condutor). Fácil! É só usar o bloco **[se ( ) então]**, passando como parâmetro o bloco **[tocando em <*Ator1*?> então]** e mudar a posição do ator no palco.

Porém, essa mudança ocorre para onde? Para cima ou para baixo? Para descobrir se vamos para cima ou para baixo, devemos fazer outro teste. Se o ator tem a posição "y" maior que zero, ele está na parte de cima e então devemos fazê-lo descer até o centro, dando um tempo diferenciado para dar o efeito de retenção dos elétrons. Para isso, para criar o 1º *loop* de teste **[se ( ) então]**, passaremos como parâmetro o bloco da abas Sensor da posição comparado com o bloco operador se é maior que zero **[(*[posição y]*) > (*0*)]***#se o valor de y está acima de zero, é sinal que o ator tocou na parte de cima#***. Para corrigir, vamos fazer uma adição com bloco **[adicione (*-1*) a y]** até cegar ao centro. Para verificar se é o centro, como não sabemos a posição, definimos que ocorra até que a condição seja satisfeita, para isso usaremos o bloco "Se Então" e como parâmetro o bloco operador lógico comparando se y é maior que "zero" **[se ((([(posição y)>([(*0*)]) então]**.

Se o ator (clone) está acima, ele deve descer e dar um tempo. Esse tempo vai dar a ideia de congestionamento, para isso use o bloco **[espere (*0.015*) seg]** *#observe que o valor é fracionado, mas use o **ponto**, não a vírgula#*. No 2º *loop* **[senão],** é só copiar o bloco para repetir com tudo dentro e mudar para adicionar 1 a "y" **[adicione (*1*) a y]**.

Figura 76 – *Script* do ator elétron



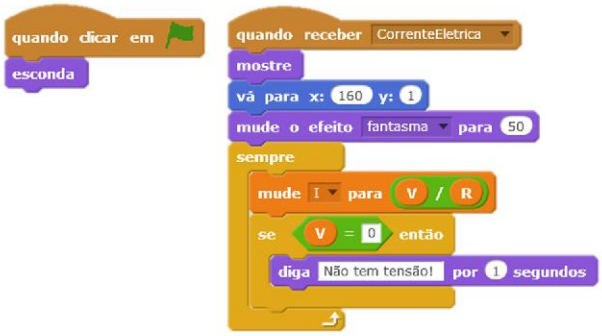
O indicador da verificação da corrente (elipse) pode ficar em qualquer posição do condutor. O *script* deve começar pelo bloco **[quando receber <*CorrenteEletrica*>]**, logo abaixo o bloco **[mostre]**, já que estava oculto.

Outro efeito é o bloco fantasma, para dar um visual de não ser materializado, sugiro com 50% do efeito **[mude o efeito <*fantasma*> para (*50*)]**.

Verificar continuamente o valor da corrente, então, use o *loop* para dar continuidade com o bloco **[sempre]** e dentro do *loop* o bloco da variável **[mude <*I*> para ( )]**passe como parâmetro o valor da equação de Omh com os blocos de operação aritmética e as variáveis **[(*[V]*)/(*[R]*)]**.

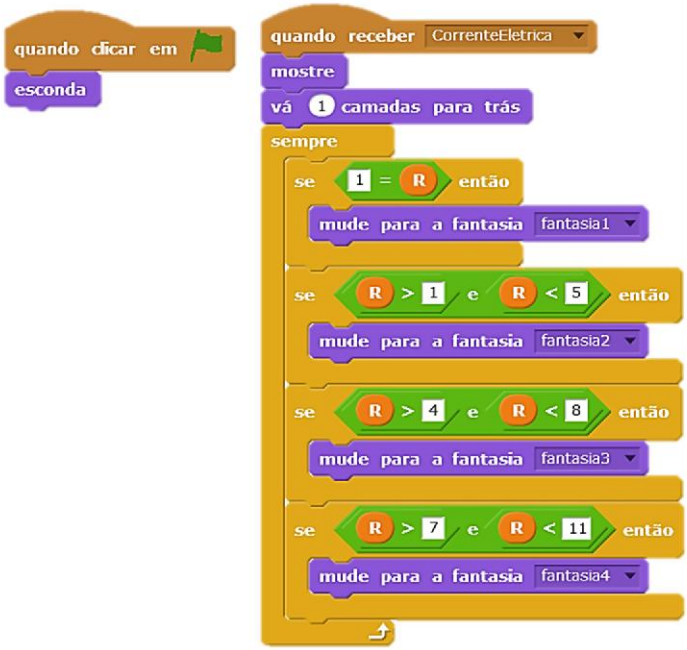
Para ficar mais coerente, vamos testar se a tensão existe, usando o bloco de teste **[se ( ) então]** e como parâmetro o bloco lógico "=" e a variável "V" **[(*[V]*) = (***0***)]**e vamos dar uma mensagem, usando o bloco de texto. Dentro do *loop* coloque o bloco **[diga (*Não tem tensão!*) por (*1*) segundos]**.

Figura 77 – *Script* da elipse



O condutor também usará o bloco **[quando receber <*CorrenteEletrica*>]**, logo abaixo o bloco **[mostre]**. Abaixo a posição no palco de uma camada atrás, para que as variáveis fiquem à frente.

Figura 78 – *Script* do condutor



Atente que a fantasia é definida pelo valor da resistência. Isso indica que temos um teste sempre, por isso os testes devem ocorre no *loop* do bloco **[sempre]**. Os testes são simples. Como fizemos apenas quatro fantasias, faremos quatro testes. Para a fantasia 1, o valor de R deve ser igual a 1, ou menor que 2. Para a fantasia 2, vamos criar um intervalo para R, para isso

usaremos o bloco lógico "e" [( ) e ( )]e dentro de cada parâmetro, passaremos o bloco, comparando o valor de R (resistência) entre 2 e 4, ou seja, se "R > 1" e "R< 5", assim faremos para as demais fantasias. Para a Fantasia 3, o valor de R fica entre 5 e7 e para a Fantasia 4, o valor de R fica entre 8 e 10. Ressalta-se que o valor no bloco é menor e maior imediatamente que o intervalo, pois o operador lógico **">"**, não inclui o valor comparado, ou seja, quando o bloco [(*[R]*) > (*1*)], quer dizer quando "R" for 2, 3... o valor "1" está excluído nessa lógica matemática.

Analisando as informações dos *scripts*

Então conseguimos fazer todas as etapas. Visualizaram que quando criamos um clone, ele concebe toda a informação de quando criado? Por isso devemos cuidar para que cada novo clone cumpra sua rotina.

Podemos colocar um ator na frente de outro, conforme a necessidade, assim como podemos alterar para frente e para trás, durante o *script*, caso seja interessante.

O uso dos blocos não está restrito ao que vemos aparentemente, mas conforme podemos combinar os blocos, podemos fazer mudanças. Porém é necessário saber "o quê" e "como", ou seja, o que se tem com resultado do bloco e como se comporta esse bloco.

Não sei se observaram, mas não finalizamos uma forma de chamar o evento da equação de Ohms. Pois bem, essa é a tarefa da semana: criar um menu e chamar os dois eventos separadamente. Na realidade, já fizemos isso no projeto 4.

Dica: no projeto 4, mudar o pano de fundo e criar um monitor de quando estamos na tela de menu. Esse monitor quando for acionado deve ocultar todos os eventos. No *script* do menu em barra usamos o "Open" e "Close".

Vídeo deste projeto

https://youtu.be/1tqjbNveni4

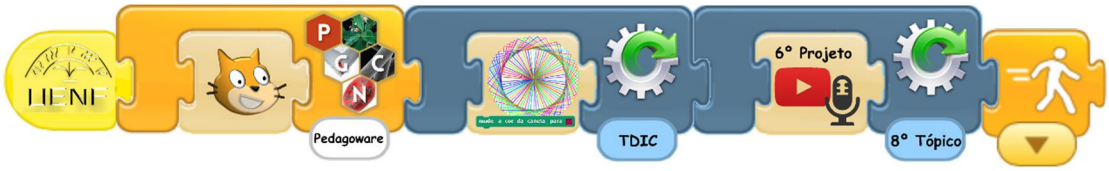
Com Scratch, podemos criar inteligência computacional

*Ao definir que um ator mude de direção quando encontrar o outro: **[tocando em <Ator1?> então]**, o que estamos fazendo é criando uma inteligência computacional para o projeto, que pode acontecer a qualquer momento. Também fizemos isso quando no Projeto 2 definimos o uso do bloco **[<direção> do vídeo em <esse ator>]**, com esse bloco qualquer que seja a ação no vídeo o sistema tomará a decisão. E quando deixamos uma orientação para quando o ator for solicitado como bloco **[quando este ator for clicado]**, e por aí vai. Na verdade, deixamos as instruções, mas quando serão executadas não sabemos, pois são ações que não temos controle.*

É possível fazer mais? Sim, vai depender de você e da sua criatividade sempre. Vamos pensar em deslocar um ator em uma pista, como fazer para ele se deslocar pela pista, nos casos de curva? Vamos considerar também que o usuário pode mudar a pista, como ficaria? Um pouquinho de criatividade e pronto. Observe as imagens e pense.

Pista (ator) ***Joaninha (ator)*** ***Bloco de decisão***

Você observou que a joaninha tem um nariz rosa comprido? E que cada pata tem uma cor? Considerando que independente do formato, a pista é padrão (preta e amarela), é simples, vou definir em uma série de testes em um loop contínuo: ***[sempre]****;* ***[se ( ) então, senão]****. Dentro do teste, vou verificar se a cor do nariz tocou na cor amarela da pista. Caso ocorra, define o movimento, se não texto se a cor que tocou é vermelha ou azul, defino que gire e mova meio passos:* ***[a cor tocando na cor]; [mova (2) passos]//[a cor tocando na cor]; [gire para esquerda (5) graus]; [mova (0.5) passos] // [a cor tocando na cor]; [gire para direita (5) graus]; [mova (0.5) passos].*** *Para garantir, faço um teste, que se tocar na borda volte, caso der algum erro.*



7º Tópico: Vinhetas

A finalidade dessa etapa é aprender a criar efeito de transição, criar bloco personalizado e dar efeitos visuais.

6º projeto: efeitos com a caneta e bloco personalizado

Até aqui vimos como movimentar, esconder, ocultar, aparecer, controlar as estruturas e variáveis, aproveitar um arquivo pronto, criar um menu, alterar o pano de fundo, conforme ocorrem eventos, copiar e duplicar *script*, *spriter* e fantasia, interagir com a câmera, com a caneta e outras coisas mais simples, ou seja, estamos prontos para começar a desenvolver um projeto de simulação. Mas para que comecemos um projeto profissional, falta uma inicialização que indique o "dono intelectual".

Já observou que todos os vídeos no YouTube ou no início de um aplicativo têm sempre uma forma de identificar o autor, que o aprecie logo no início? Pois bem, esse formato virtual, que é também uma arte, é denominado de vinheta.

A vinheta produz uma identidade, por intermédio de recursos de áudio e visual, para o mundo socialmente líquido e fluido saber que existe algo ou alguém por "trás" daquele conceito. Nesse processo podemos considerar que a vinheta é uma forma eletrônica de estar presente quando sua intelectualidade alcança diferentes "distâncias físicas" por intermédio dos ambientes virtuais.

Sendo assim, vamos criar uma vinheta, um projeto com uma arte visual. Escolhi a aba de caneta, que já vimos um pouco, e a aba "Mais blocos", ou como prefiro: Blocos personalizados. Vamos explorar com várias formas, mas com certeza não vamos exaurir os recursos.

Como já conhecemos a caneta (3ª projeto) vou começar a fazer algumas etapas e depois vou unificar. Isso é possível pois podemos personalizar um bloco usando a aba "Mais blocos".

Inicialmente apague o ator "Gato" e crie um novo ator; assim como no 3º projeto, não haverá fantasia (nem um ponto).

Script para criar uma espiral

Bloco para criar uma espiral, tipo um caracol crescente, que se desenvolve na tela. Usarei a tela de palco preta, pois facilitará a visualização de todas as cores.

Figura 79 – *Fases de apresentação do script na tela*

Inicialmente iremos criar um bloco personalizado. Acessando a aba "Mais blocos", será apresentada uma aba, passo (1) da Figura 80. Na aba é só dar um *click* dentro do bloco e digitar o nome do bloco, após clicar em "ok". Quando for confirmado, aparecerá o bloco especial (2)

com a definição do *script* e o bloco de duplo encaixe (3), que poderá se encaixado durante o *script* principal e acionar esse bloco criado. Na área de computação (programação), denominamos de orientação aos objetos, pois posso criar um objeto (bloco) e acioná-lo em qualquer parte do algoritmo quantas vezes quiser sem ter que repetir; além do mais é bem simples para organizar e não deixar um *script* muito grande e confuso.

Figura 80 – Fases de apresentação do script na tela

Vamos precisar de uma variável para controlar a posição da caneta no evento, para isso vamos criar o bloco variável **[Circunferencia]**.

Usando o bloco **[defina <*Espiral*>],** vamos inicializar a variável com valor zero. Use o bloco **[mude <*Circunferencia*> para (*0*)]**; garanta a fantasia (ao longo do projeto entenderá, pois usaremos outras fantasia nesse mesmo ambiente), use o bloco **[mude para a fantasia <*Costume1*>]** #*"Costume 1" é o nome que dei a fantasia sem ator. Explicarei mais adiante porque definir a fantasia* #; nesse caso, estou garantindo a cor que iniciará a circunferência, por isso usei o bloco **[mude a cor da caneta para <azul>]**; depois vamos garantir que não haverá marcas no deslocamento da caneta; usaremos o bloco **[levante a caneta]**; vamos definir o tamanho da caneta com o bloco **[mude o tamanho da caneta para ( )]***#sugiro o valor para 8#*; defina a posição inicial com o bloco **[vá para x: ( ) y: ( )]***#para o tamanho que está planejado a sugestão é a coordenada (0,0)#;* e usar a caneta com o bloco **[use a caneta]**. Até aqui nós conhecemos, praticamente é o que fizemos no projeto 3. O que é diferente é o *loop*. Relembrando o que usamos no projeto 3, o *loop* repita, aqui será a mesma coisa. Mas o que desejo é riscar (marcar com a caneta) do centro para fora até próximo da borda.

Vamos colocar nesse *script* um bloco de controle de repetições **[repita ( ) vezes]***#a sugestão do efeito na minha tela é que faça isso por* ***650*** *vezes#*. Dentro do *loop* de repetição vamos colocar uma incrementar a variável que criamos com o bloco **[adicione a <*Circunferencia*> ( )]** *#estou sugerindo o valor de 0.05.* ***Observe novamente o uso do ponto, não de vírgula****#*. Agora vamos fazer a caneta deslocar-se no palco com o bloco **[mova ( ) passos]***#o parâmetro nesse caso é o bloco que criamos **[Circunferencia]**#*. Agora vamos girar usando o bloco **[gire seta para direita ( ) graus]***#estou sugerindo o valor de 12°#*; para finalizar, vamos mudar a cor da caneta, usando o bloco **[adicione (*1*) a cor da caneta]** *#Note que uso o adicione, isso fará uma mudança mais suave e não mude. Se usar o bloco mude, sem uma incrementação, ficará preso na cor indicada pelo número de parâmetro#.*

Figura 81 – Tela e *script* da circunferência

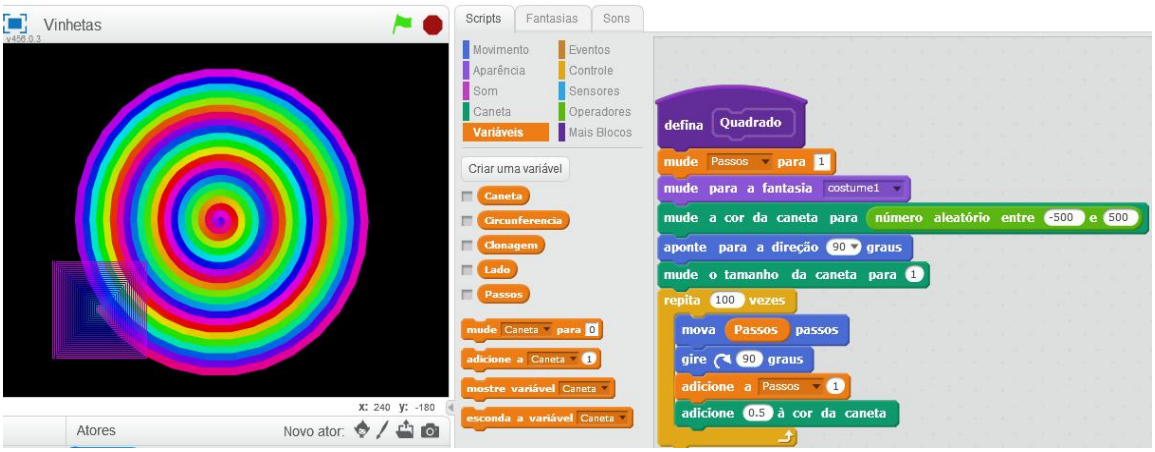
Relembrando o Projeto 3, o que fizemos aqui é muito próximo, ou seja, é só uma nova forma criativa de ver o uso da caneta e, assim, vamos continuar para atender o resto do projeto.

Vamos criar outro *script,* com outro bloco personalizado, para criar um quadrado. Porém vou criar um quadrado da posição que pare a circunferência e fazer esse quadrado sobre a circunferência.

Os passos iniciais são os mesmos: Criar um bloco que estou nominando de quadrado, criar uma variável para garantir a evolução, que denominei passos. No bloco **[defina *<Quadrado>*]** inicialize a variável com o bloco **[mude *<Passos>* para (*1*)]**, garanto a fantasia com o bloco **[mude para fantasia *<costume1>*]**; defino a cor de forma aleatória com o bloco **[mude a cor da caneta para ( )]***#passei como parâmetro o bloco **[número aleatório entre ( ) e ( )]** e preenchi com valor bem amplo entre -500 e 500#*; defini a direção com o bloco **[aponte para a direção (*90*) graus]**, isso é preciso, pois, estamos fazendo um quadrado; eu quero que o novo desenho faça uma sobreposição de imagens e não que apague a imagem anterior, por isso defini a caneta para 1 com o bloco **[mude o tamanho da caneta para (*1*)]**. Agora que preparamos vamos criar o *loop*.

Como sempre um *loop* de repetição que usei 100 vezes. Dentro do *loop* defini o movimento com o bloco **[mova ( ) passos]***#passei como parâmetro o bloco que criamos **[Passos]**#*; determinei o giro em 90 graus, com o bloco **[gire para direita (*90*) graus]***#relembrando: 90º pois é um quadrado e quero que seja apresentado na tela em referência ao plano cartesiano#*; incrementamos a variável com o valor "1" usando o bloco **[adicione a Passos (*1*)]**; por fim mudei a cor da caneta durante a construção, porém, dessa vez mais lento com o bloco **[adicione (*0.5*) à cor da caneta]***#mais uma vez, atente que além do parâmetro de valor "0.5", também usei o bloco **adicione** e não o bloco **mude a cor**#.*

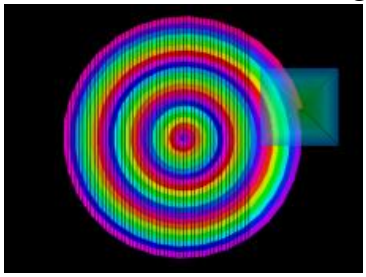
Figura 82 – Tela e *script* da circunferência e o quadrado na sequência.

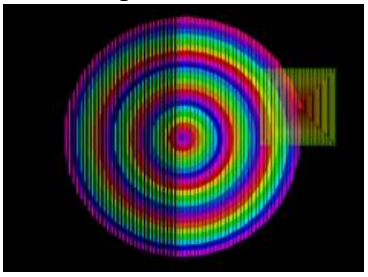


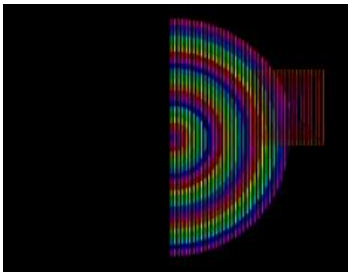
Olhe o que aconteceu: além de criar um quadrado, que se constrói sobrepondo a imagem anterior, assim como no processo da circunferência, criamos uma forma de mudança de cor, porém dessa vez mais suave.

Bem, agora nossa tela está com algumas informações, e precisamos ir para mais uma etapa, então vamos criar uma transição que apague esse efeito. Mas não quero simplesmente apagar, quero criar uma suavidade como é demonstrado na Figura 83.

Figura 83 – Etapas do efeito de transição


Tela iniciando a transição


Tela média de transição


Tela finalizando a transição

Observe que as linhas pretas vão ocultando a imagem criada. Escolhi preta, porque também escolhi o fundo preto.

Para esse efeito, criei o bloco na aba “Mais bloco” com o nome “Transição” e uma variável de nome caneta. Abaixo do bloco **[define <*Transição*>]**, inicializei a variável pra “1”, com o bloco **[mude <*caneta*> para (*1*)]**, como queria em apenas três passagens, criei um *loop* de repetição de três vezes com o bloco **[repita (*3*) vezes]**. Dentro do *loop* defini que a caneta seria levantada, para não marcar a tela com o bloco **[levante a caneta]**; defini a cor para preto com o bloco **[mude a cor da caneta <*preto*>]**; mandei o cursor para a posição onde queria iniciar com o bloco **[vá para x: (*-240*) y: (*0*)]**; defini o tamanho da caneta com o bloco **[mude o tamanho da caneta para ( )]***#passa como parâmetro o bloco criado “Caneta”, que irá ser aumentado o valor#*. Pronto, defini todas as etapas e agora, dentro de outro *loop,* vamos fazer essa linha percorrer a tela da esquerda para direita de quem observa a tela.

Criei um *loop* com 100 repetições, que será o suficiente, movimentando 5 passos no eixo “x” como o bloco **[repita (*100*) vezes]**, esse *loop* fica dentro de outro *loop* de 3 vezes. Dentro desse novo loop defini que a ordenada “y” vá para a posição 240, “x” mova 5 passos e “y” volte para a posição -240, com os blocos **[mude y para (*240*)]**, **[adicione (*5*) a x]** e **[mude y para (*-240*)]**.

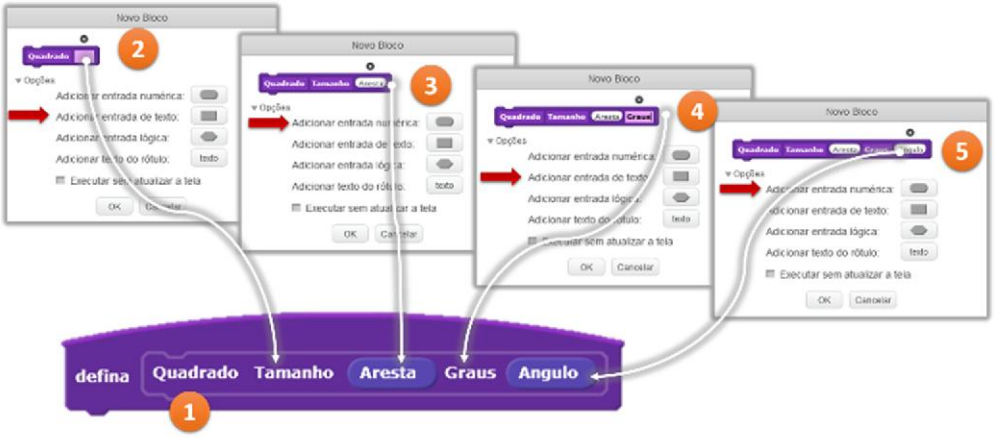
Ainda dentro do *loop* de três repetições e fora do de 100, incremente a variável “Caneta” com o bloco **[adicione a <caneta> (*2*)]**.

Figura 84 – Etapas do efeito de transição

O que esses *loops* fazem é deslocar a caneta de cima para baixo por cem vezes incrementa x em 5, depois aumenta a espessura da caneta em 2 e repete por mais duas vezes. Assim, temos que na primeira passagem vai marcar 1 unidade de cima para baixo e evoluir em 5 unidades de x, ou seja, temos 4 unidades visualizando a imagem que criamos {↓↓↓↓↓↓↓↓... *#a seta preta representa o risco na tela e a seta cinza a parte da tela que está aparecendo#*}. Na segunda passagem, retorna para o início, faz o mesmo percurso, só que com a caneta valendo o valor da adição (1 + 2 = 3), assim o efeito é {↓↓↓↓↓↓↓↓...}. Repete-se esse efeito, só que a caneta vai valer agora o total do espaço (3 + 2 = 5), por isso apaga totalmente {↓↓↓↓↓↓↓↓↓...}.

Agora vamos utilizar o mesmo *script* com que desenhamos os polígonos, só que dessa vez vamos usar o bloco que vou denominar de "Quadrado"; vou criar o bloco quadrado, mas vou chamar esse bloco mais de uma vez e passando mais de uma informação. Observe a figura a seguir.

Figura 85 – Criação do bloco "Quadrado" com parâmetros



Observando a Figura 85, o primeiro passo (1) foi nominar o bloco "quadrado". Depois, como efeito didático para qualquer um saber, adicionei o texto "Tamanho" (2), seguido do valor do tamanho do quadrado com o nome "Aresta" (3), mais um texto "Graus" (4) e o valor numérico de graus com o nome Ângulo (5). O bloco poderia ser criado só com os valores,

porém poderia ficar confuso para qualquer um, inclusive para mim (que escrevi o *script*), quando quisesse rever o *script*.

Esse tipo de bloco suporta texto, valor numérico, valor lógico e *string* (caractere), basta criar o que precisa, para o caso apenas dos textos de orientação e dois valores numéricos: tamanho do quadrado e ângulo de execução.

Abaixo do bloco, apenas voou definir a fantasia como bloco **[mude para a fantasia *<costume1>*]**. Vou criar um *loop* para fazer repetir em 40 vezes com o *loop* **[repita (*40*) vezes]**. Sei que o correto seria fazer apenas 36, 72... (múltiplos de 360), mas, estou fazendo 40 para garantir que não vai haver erros. Dentro do *loop*, como é um quadrado, vou colocar outro *loop* já conhecido **[repita (*4*) vezes]** e dentro desse bloco dois blocos para criar o quadrado com os blocos: **[mova ( ) passos]***#passo o valor "Aresta", pois assim o que vier com valor nessa posição vai funcionar. Para passar "Aresta" como valor, é só arrastar o próprio bloco defina, e quantas vezes arrastar o bloco vai parecer outro#* e o bloco **[gire seta para direita (*90*) graus]**.

Abaixo, defino a volta para a posição na coordenada (0,0) e um novo giro com os blocos **[vá para x: (*0*) y: (*0*)]** e **[gire seta para direita ( ) graus]***#vamos passar como parâmetro o valor do bloco [Angulo], que nesse caso sai do próprio bloco personalizado que criamos#*. E por fim, só para dar um charme, coloquei um *delay* com o bloco **[espere (*0.09*) seg]**. Esse *delay* é porque às vezes esse tipo de bloco pode aparecer pronto na tela, não se construir lentamente.

Figura 86 – *Script* "Arranjo"

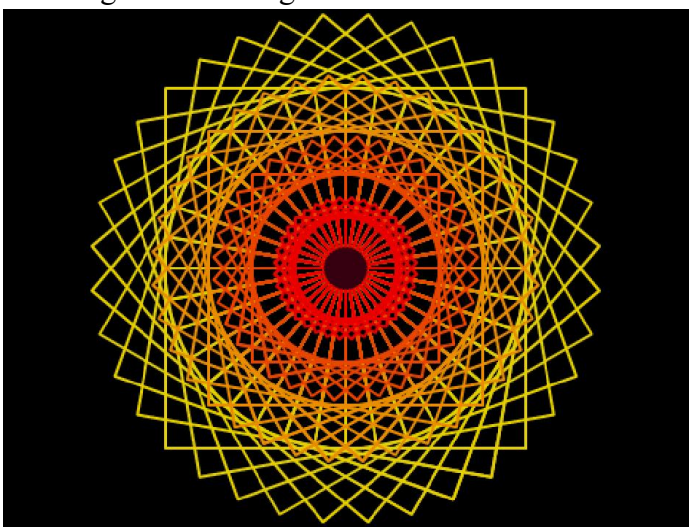
Bem, agora vamos fazer esses blocos serem "chamados" na programação. Com o bloco **[quando clicar em ⚑]**, vamos empilhar os blocos, inicialmente garantir posição e limpeza da tela quando iniciar, então usemos os blocos **[levante a caneta]** e **[apague tudo]**, agora é só chamar os blocos na ordem que deseja, estou sugerindo o bloco **[Espiral]**, **[Quadrado]** e **[Transição]**.

Simples assim, nós executamos os blocos e estão fácies de entender. Mas lembre-se de que no último bloco não houve cuidado para montar a posição do *script*, pois agora terá que ser feito, então antes de chamar o *script* para os polígonos, vamos preparar o palco. Inicialmente apagaremos os efeitos, definiremos a caneta, levantaremos a caneta para não marcar o palco e determinar a posição da caneta, essas etapas nós já conhecemos, mas caso haja dúvida o script está abaixo do texto.

Vamos criar uma variável "Lado" e iniciar com o bloco **[mude *<Lado>* para ( )]** *#estou sugerindo o valor 125#*, definir a cor da caneta inicial com o bloco **[mude a cor da caneta**

para < *"amarelo limão"* **>]**. Agora já está tudo preparado é só acionar o bloco, porém eu quero que uma circunferência de polígonos crie outra interna, como se fosse um girassol.

Figura 87 – Imagem do Girassol estilizado



Para isso vamos criar um *loop* com quatro chamadas ao bloco quadrado. Com o *loop,* repita por 4 vezes. Dentro do *loop,* vamos passar o bloco "Quadrado", porém esse bloco espera dois parâmetros, um será o bloco variável **[Lado]** e o segundo parâmetro será o valor 10. Ainda no loop vou redefinir a cor da caneta para menos dez tons como bloco **[adicione (*-10*) à cor da caneta]** e vou incrementar o valor de Lado para 30 negativo como bloco **[adicione a <*Lado*> (*-30*)]**. Porque o valor negativo? O valor negativo é porque desejo criar o arranjo de fora para dentro. E as cores serão definidas conforme as especificações.

Já para o miolo desse "girassol estilizado", escolhi uma tonalidade específica e um tamanho fora desse padrão; então é preciso passar novamente o bloco de cor **[mude a tonalidade da caneta para (*3*)]** e o bloco **[Quadrado tamanho (*10*) Graus (*10*)]**.

Bem, agora vem o "pulo do gato". Eu poderia fazer o movimento desse gráfico, contudo isso demanda uma linha de pensamento algébrico complexo, então vamos criar uma fantasia. Após montado o nosso girassol estilizado, vamos dar um *print* na tela, colar esse *print* em um *software* qualquer (pode ser o PowerPoint) e retirar as informações que não desejamos.

Vamos mandar apagar toda a tela com o bloco **[apague tudo]** e colocar a segunda fantasia no palco, definir que a fantasia seja vista como bloco **[mostre]**, trocar a fantasia com o bloco **[mude para a fantasia <*Capture*>]** *#capture é a fantasia que criamos#*; coloque no tamanho desejado para ficar do mesmo tamanho que a imagem desenhada (eu usei 100%), como o bloco **[mude o tamanho pra (*100*) %]**, vamos dar um efeito para retirar a imagem do palco.

O efeito que escolhi foi diminuir e olho de peixe. Para isso criei um *loop* de 120 repetições, diminuindo 1 sempre e adicionado o efeito olho de peixe com valor 2. Observem o *script*.

Figura 88 – *Script* para desenvolver os blocos

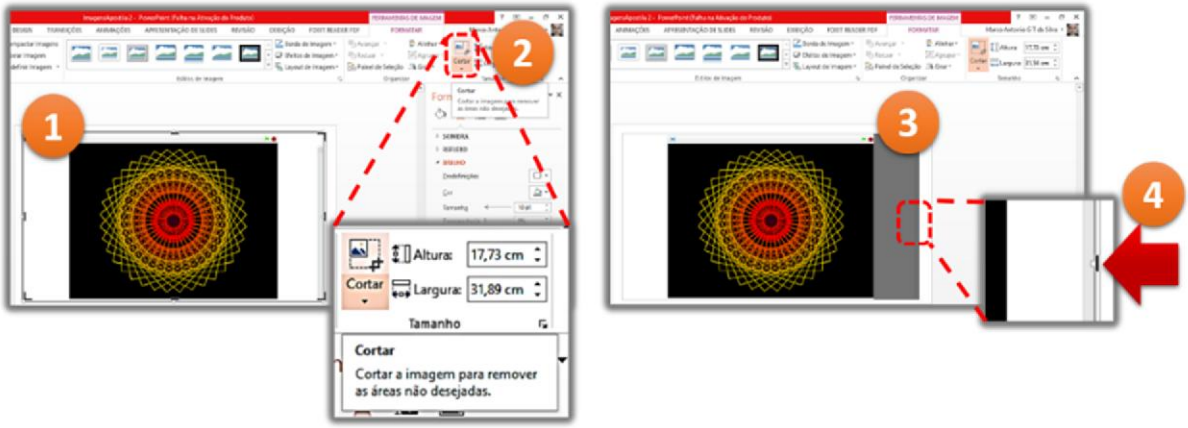
No final desse *script* enviamos uma mensagem. Essa mensagem é para criar um efeito para o mouse. Como retiramos o ator do palco por efeito gráfico, vamos esconder como bloco **[quando receber <*FinalizarVinheta*>]** e o bloco **[esconda]**.

Como criar um spriter no PowerPoint (ou limpar uma imagem)?

Para editar a imagem no PowerPoint, observe a figura: o primeiro passo (1) é selecionar a figura, ir até a aba "Formatar" e selecionar a ferramenta cortar (detalhe do passo (2)), clicar no detalhe do passo (4), uma linha mais grossa que aparece nas laterais e arrastar até a posição que deseja (3).

Depois dessa etapa, salve a figura como png ou jpeg. Eu salvei como o nome "Capture". Agora no Scratch, na aba fantasias do spriter, na ferramenta da pasta aberta com a seta para cima ( ), carregue o traje a partir do computador: vai abrir uma janela de diálogo; é só selecionar a imagem criada no formato "jpeg", ou "png".

Figura 89 – Limpando a imagem no PowerPoint com a ferramenta cortar (tesoura)



Para encerrar, criaremos um efeito para o mouse

Em outro *spriter*, criei uma pequena bolinha, que estou usando na cor vermelha, e defino que quando o projeto iniciar o ator deve ficar oculto. E, apenas quando receber a mensagem "FinalizarVinheta", defino que deve mostrar, apagar qualquer efeito, mudo o efeito cor para valor aleatório entre 1 e 500 e o tamanho para 50% (vai depender da fantasia que criou, para o que queria, precisei deixar com 50% do tamanho). Criei um *loop* sempre, nesse *loop* defino que o ator vá para a posição do mouse, como está no *loop* sempre isso é atualizado todo instante e qualquer movimento do mouse adiciona o efeito cor, cria um clone, só que não desejo encher a tela de bolinhas, por isso eu crio o clone a cada "0.03" segundos.

Quando o clone for criado, mudo a cor de forma aleatória entre -500 e 500, mudo o tamanho para 75% (maior que o ator que está sempre na tela). Crio outro *loop* repetindo 10 vezes, nesse *loop* mudo a cor, dou efeito fantasma e diminuo o tamanho, gero um movimento rotacional com dez passos e mando deslizar para a posição do mouse, em um tempo maior que o ator. Essa rotina do *loop* vai dar o efeito de que estão sempre seguindo o mouse, porém parece que perdem a referência e estão sempre buscando.

O efeito fantasma dá uma transparência, juntamente com a troca de cor parece que as luzes estão piscando durante o deslocamento.

Figura 90 – *Script* para dar efeito no mouse

Figura 91A – Efeito na tela

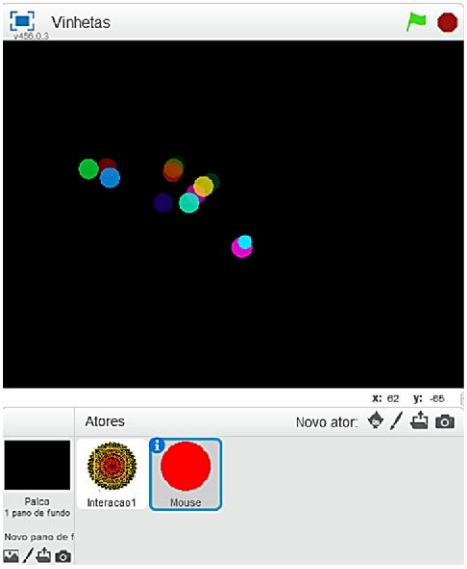
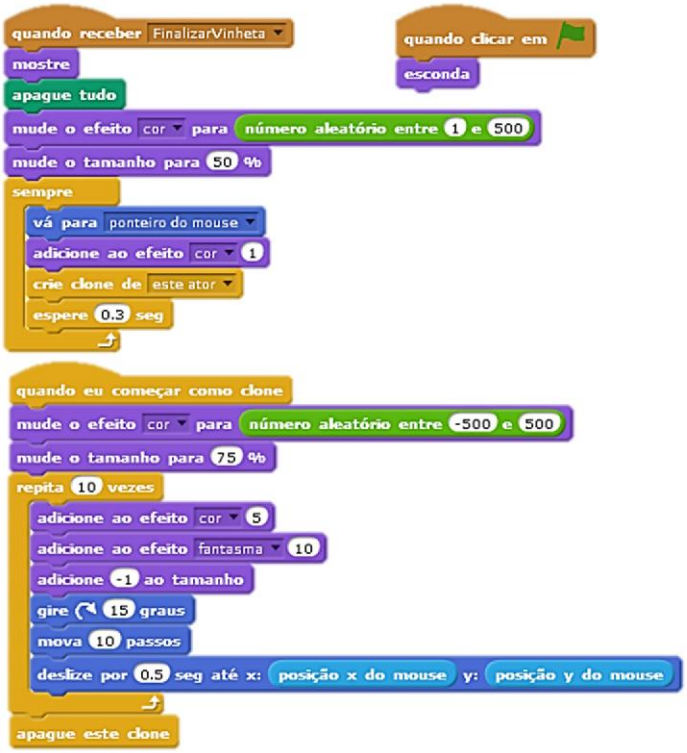


Figura 91B – Script



Análise do *script*

Vimos vários recursos com os blocos personalizados. Quando tratamos tudo dentro do bloco ficou mais fácil trabalhar o *script* para chamar, porém quando não fizemos, tivemos que tratar tudo antes, como foi realizado no bloco do girassol estilizado. Outra constatação interessante é que se houvesse a necessidade de criar cada efeito girassol, teríamos um *script* muito grande, que seria recriado cinco vezes. Com o bloco personalizado, ficou mais fácil e o *script* principal ficou mais limpo.

É claro que existem vários outros recursos, se forem à página do Scratch e buscarem por "pen", "art" ou "fractal" vão achar ótimos *scripts*, alguns mais complexos, mais belos, mais empolgantes, outros mais simples, ou seja, são várias formas de utilizar.

Outra coisa, vocês observaram que no último *script* eu não disse mais qual era o bloco que estava sendo usado? Já chegamos a uma fase mais prática e temos algum conhecimento. Daqui para frente vamos trabalhar mais nesse formato que é a sintaxe "portugol".

Bem, nosso objetivo é criar uma vinheta e deixamos vários formatos, você ainda pode achar outros diferentes também, mas observe que uma vinheta tem uma música sempre. Esse é o nosso desafio: criar uma vinheta com música de fundo, que poderá ser a partir desses exemplos ou outros que preferir. Aguardamos sua vinheta.

Não esqueça que uma música tem direitos autorais, porém existem sonorizações livres de direitos autorais (*royalties free*). O próprio YouTube possui um acervo de músicas e efeitos sonoros livres de direitos autorais <https://www.youtube.com/audiolibrary/music>.

Vídeos deste projeto

Script do projeto

https://youtu.be/aEkdEWIs6yc

Criação de ator no PowerPoint

https://youtu.be/OAcDMicH5wU

Dividir para conquistar!!!!!!

O que o bloco personalizado faz? Na verdade, assim como os blocos de mensagens, o que fazemos é dividir o script em procedimentos (na área de programação é definido por procedures). Porém os blocos personalizados são do ator para o qual o bloco foi criado, já com referência à mensagem, é possível passar uma mensagem para todos os atores.

No bloco personalizado, assim como na variável, ao serem criados surgem todas as instruções para manipular os dados.

O funcionamento é bem simples: o script em execução envia a mensagem e essa mensagem dispara uma ação. Esse evento é definido na área de computação como gatilho (trigger). Para entender melhor, seria como se ao chegar à execução do bloco, esse bloco mandasse uma mensagem para o outro; quando for uma mensagem, vai para todos (broadcasting), quando for um bloco personalizado, fica somente no spriter a mensagem, caso seja um bloco envie para todos e a mensagem chegará para todos os atores. Em todos os casos, os blocos estão sempre esperando para serem executados.

No caso do bloco personalizado, existe a opção: "Executar sem atualizar a tela". Essa opção faz com que todos os blocos executem sua instrução e apresentem na tela o efeito final. Isso aumenta a velocidade.

8º Tópico: Material potencialmente significativo

A finalidade dessa etapa é discutir o planejamento do recurso de TDIC com base em uma teoria educacional.

Desenvolvimento de material digital potencialmente significativo

Ao iniciar na área de desenvolvimento de aplicativos ou interações computacionais para educação, existem vários requisitos a serem observados: métricas computacionais, desempenho cognitivo do objeto (aplicativo) e as questões das técnicas de programação.

Sobre as métricas, vamos observar a relação histórica que o *software* ou aplicativo sempre foi desenvolvido por profissionais da área de informação. Mesmo os aplicativos com fins educacionais não eram desenvolvidos por educadores. A questão de esforço computacional e padronizações, no final da década do século passado, levaram a várias normatizações, incluindo mensurações para quantificar a tendência de como avaliar o desenvolvimento do sistema computacional.

Rocha e Campos publicaram, em 1993, que é necessário avaliar o *software* e estabelecer métrica de qualidade que incluam principalmente (re)usabilidade, confiabilidade e portabilidade. Essas mensurações deveriam avaliar apenas o *software* e o conhecimento produzido pelo *software* deveria ser avaliado pelos professores. Contudo, Rocha e Campos (1993) definem que o profissional da educação não precisaria desenvolver *softwares*, apenas saber avaliar e utilizar o produto da área computacional. Esse discurso possivelmente desenvolveu uma mentalidade que os professores não desenvolvem e, consequentemente, não criam "intimidade" com as tecnologias entrantes da área de programação.

Na década seguinte, a preocupação com o desenvolvimento de *softwares* educacionais chega ao Ministério da Educação e é criada "A Rede Internacional de Educação Virtual" (ou *Red Internacional de Educación Virtual para el Mejoramiento del Aprendizaje en Ciencias y Matemáticas en América Latina*) – RIVED. O Rived possui um foco mais educacional e menos computacional.

Mais atual, o professor José Armando Valente (2014), já definindo as tecnologias no ambiente digital, relata que as Tecnologias Digitais de Informação e Comunicação (TDIC) na educação não promovem mudanças substanciais na sala de aula com o mesmo potencial que envolve a sociedade.

É nítida a preocupação com o tipo de recurso da TDIC a ser aplicado na educação. Por isso, é necessário saber o que vamos fazer e também se essa preocupação talvez tenha criado um *layout* em que o professor não era o criador do *design* da tecnologia a ser aplicada em sala.

É necessário saber que a área de programação é útil ao docente, mas esse profissional não tem formação para o desenvolvimento de projetos de programação que permita reproduzir os comportamentos dos fenômenos das ciências.

Outro ponto é: E o aprendiz? Como ele se comporta com o *software* educacional? Em geral esse conceito não é relatado, pois quando se desenvolve um aplicativo, este está voltado para as métricas computacionais e possivelmente para as orientações educacionais.

No que diz respeito ao aprendizado, é coerente ressaltar que, ao passar pelas disciplinas, ele deve codificar conhecimentos prévios, que são acumulados, mesmo que esses

conhecimentos encontrem-se num estágio pouco aprofundado. Portanto, é coerente recorrer ao conceito de aprendizagem proposto por David P. Ausubel (2003), segundo o qual os subsunçores[3] pré-existentes na estrutura cognitiva do indivíduo tendem a ser utilizados como recursos didáticos, para mediar a estruturação do conhecimento.

A aprendizagem significativa é definida por Ausubel (2003), quando uma proposição[4] lógica se relaciona na estrutura cognitiva do aprendiz. Nesse processo, a aprendizagem tornar-se subordinada correlativa, pois há uma extensão ou qualificação de proposições (ideias relevantes) já aprendidas anteriormente e contextualizadas, não é apenas derivativa, pois transpõem a exemplificação (MOREIRA, 2006).

Destaca-se então que a teoria ausubeliana, da aprendizagem significativa, faz-se necessária quando se destaca o desenvolvimento de material potencialmente significativo. Ou seja, a construção de material com recursos da tecnologia pode e deve abordar os conhecimentos preexistentes na cognição do aprendiz, favorecendo para que o novo conhecimento seja fixado na estrutura cognitiva existente.

Nessa configuração, o conteúdo sequencial do ensino desempenha, portanto, a função de organizador relevante para aprendizagem, sendo necessário ser acordado e torná-lo conteúdo "fresco". Assim, se realiza uma prática com finalidade distinta e voltada ao contexto de interesse (AUSUBEL, 2003).

Ausubel (2003) afirma que a aprendizagem significativa e a assimilação do conteúdo incluem uma ancoragem, também seletiva, dessa seleção. Ou seja, a aprendizagem não ocorre no vácuo cognitivo. Nesse caso, o material que será usado para instruir o processo, seja digital ou não, o qual é definido por Ausubel (2003) e Moreira (2006), como material potencialmente significativo, deve observar e se relacionar com os conceitos existentes na cognição e a nova estrutura do aprendiz (MOREIRA, 2006). Assim sendo, é necessário desenvolver materiais que acordem a "cognição" do aprendiz.

Ressalta-se que na teoria ausubeliana o material potencialmente significativo não retrata apenas sequências de informações, mas não é arbitrário em si mesmo ou nas suas correlações.

O emprego da tecnologia, dentro do propósito metodológico desde seu planejamento, apresenta melhorias significativas no processo ensino e aprendizagem de áreas que dependem de ensino de concepções abstratas. De tal modo, é adequado aplicar a tecnologia fundamentada em uma metodologia planejada, para que essa produza o efeito desejado e não seja apenas um formato diferenciado do livro didático.

Por isso, ao desenvolver um *software* educacional, é necessário não apenas vislumbrar as métricas, que historicamente coordenaram o processo de desenvolvimento. Porém, acredito que o docente, por estar inserido no contexto educacional, seja o melhor "referencial métrico" para criar aplicativos significativos que atendam a construção de conhecimento.

Creio que o desenvolvimento de um simulador digital por um docente, está intimamente ligado à sua forma de trabalhar e assim, também possui uma confiabilidade pedagógica. Observando que o conteúdo pode ser explorado de várias formas e temos realmente uma reusabilidade, quanto às métricas computacionais, a própria linguagem Scratch propicia a (re)usabilidade, como vimos no projeto 5, a portabilidade, pois pode-se usar *off-line*, *on-line* e com diferentes plataformas, já a confiabilidade vai depender especificamente do que se pretende atingir e para que foi projetado.

Cabe então a questão do projeto: o docente deve planejar e pensar quais são os objetivos, qual seu público alvo, mapear os processos educacionais que podem ajudar para atingir os

[3] O subsunçor refere-se ao conceito ancoradouro aos pressupostos já existentes no processo de aprendizagem (MOREIRA, 2006), que induz aos novos arranjos mentais a partir das informações adquiridas.

[4] Uma proposição é a aprendizagem dos significados, onde dois ou mais conceitos compõem-se em uma semântica. "Por exemplo, a proposição referente à lei de Ohm só poderá ser aprendida significativamente depois que forem aprendidos os conceitos que, combinados, constituem tal proposição" (MOREIRA, 2006, p. 26).

objetivos, criar uma breve descrição e, nessa descrição, incluir cenário visual e contextual de cada interação visual. Isso tudo também são métricas, mas métricas da educação com material potencialmente significativo, não apenas um *software* educacional.

Vídeo do meu projeto

Essa semana o vídeo que vocês irão observar será do meu projeto, para entenderem o que estamos esperando. O projeto foi dividido em quatro partes, sendo a última uma dica.

Parte 1

https://youtu.be/4mdjjMGB4zc

Parte 2

https://youtu.be/J8P3DEz74xo

Parte 3

https://youtu.be/3mBTVyPH9PE

Construção de imagem no PowerPoint

https://youtu.be/__JtPa8s2hw

9º Tópico: Portabilidade de objeto digital de aprendizagem

A finalidade dessa etapa é publicar o projeto em ambiente virtual.

Publicação de projetos (objetos)

Como já comentado, a estrutura de um Objeto de Aprendizagem no formato digital deve expor etapas do ciclo de formação que antecedem o conteúdo do objetivo, possibilitando uma análise reflexiva sobre a experiência do aluno. Tal processo facilita a incorporação de novos constructos na cognição do aprendiz, usando, portanto, a simulação do evento servirá como fonte de construção da aprendizagem (FANNING, GABA, 2007).

O desenvolvimento de uma interface digital, mediada por intermédio de efeitos midiáticos (imagem, som e movimento) para o ensino de ciências, onde os conceitos literais são representados por mídias que representam eventos de movimento, é denominado simulação.

Um Objeto Digital de Aprendizagem (ODA), que viabiliza a simulação de um dado conceito ou fenômeno, deve considerar as características de (re)usabilidade e disponibilidade na *web*, isto é, pode ser empregada, utilizada, reutilizada e referenciada para a aprendizagem em diferentes contextos. É significante que a (re)usabilidade seja disponível em ambiente digital público, ou seja, na rede mundial de computadores (WILEY, 2000; TAVARES et al., 2007).

No contexto identificado por Wiley (2000), o objeto compõe-se em ODA por apresentar potencial para ser reutilizado e adaptável, não só pelos conceitos de algoritmos computacionais, mas também dentro do contexto do ensino e aprendizagem (WILEY, 2000).

Assim, estando o ODA disponível para execução *on-line*, torna-se independente da plataforma de sistema operacional e pode ser acessado em dispositivos móveis e computadores *desktop*, desde que haja compatibilidade do sistema.

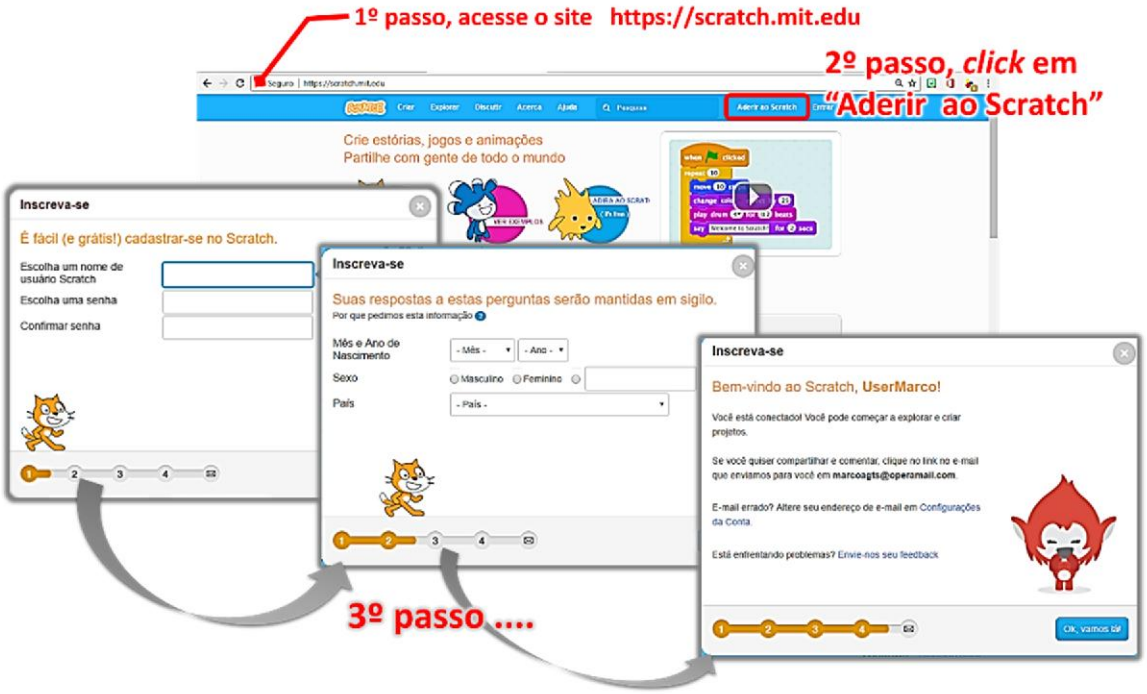
O contexto apresentado por Wiley (2000) e Tavares et al (2007) pressupõe uma interface que dê suporte à informação digital trabalhada, uma vez que na *web* nem tudo pode ser publicado.

O Scratch possui uma interface na *web* que permite colocar o projeto em andamento ou pronto. O projeto em andamento, uma vez publicado, permite uma coperação entre as pessoas. Imagine que temos uma dificuldade com um *script* e não conseguimos andar, se estiver público na *web,* podemos mandar o *link* para um colega e pedir ajuda, além de garantir a portabilidade.

Examine novamente o último parágrafo e a menção de cooperação, não colaboração, pois a "*colaboração é uma filosofia de interação e um estilo de vida pessoal, onde o indivíduo é responsável pela sua ação"* e a *"cooperação é uma estrutura de interação projetada para facilitar a realização de um objetivo ou produto final por intermédio de pessoas que trabalham em grupo*" (PANITZ, 1999).

Caso não tenha feito no início do curso, reveja como se cadastrar, para poder publicar o projeto Scratch no site do MIT. Inicialmente acesse o site do projeto Scratch (https://scratch.mit.edu) e *click* em aderir (2º passo da figura). Após essa etapa, aparecerão as janelas para preencher o nome, senha; a seguir, em outra aba, aparece a solicitação de data de nascimento, sexo e país; na última aba de dados, pede-se o *e-mail* e confirmação e por fim é apresentada uma mensagem de bem-vindo.

Figura 91 – Cadastro no site do projeto Scratch



Uma vez cadastrado, você poderá acessar diretamente o site (Figura 93A). Após acessá-lo, poderá criar novos projetos ou um ambiente para publicar e incorporar outros projetos de seu interesse, que é o "Novo estúdio" (Figura 93B).

Figura 92 – Cadastro no site do projeto Scratch

Figura 93A – Tela de acesso

Figura 93B – Tela do usuário

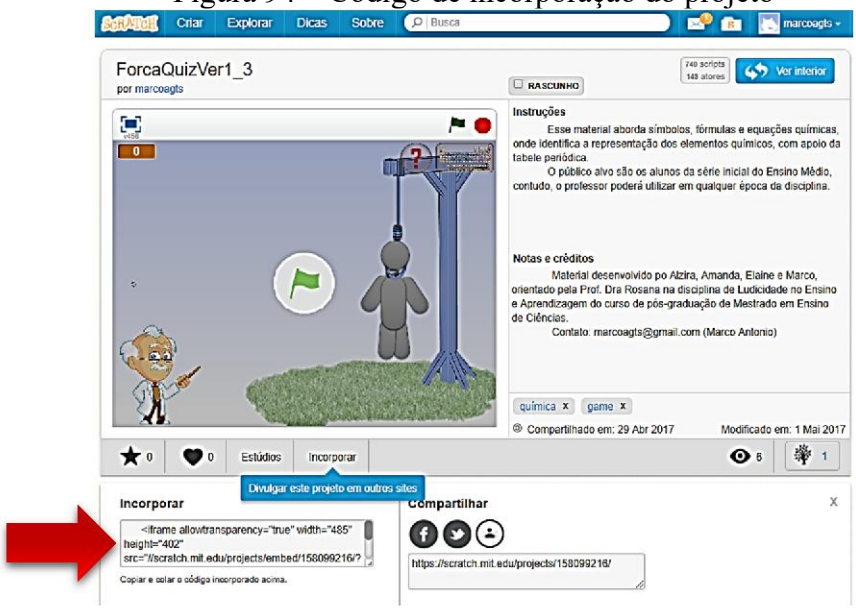
Bem, já podemos criar no site e publicar, como também enviar do seu *desktop* para o site. O envio ou a criação não garantem a publicação, apenas que estará disponível nas nuvens, para ficar público é necessário clicar no botão "Compartilhar", no canto da tela superior direita de quem observa.

Para enviar do *desktop,* é necessário o *login* e a senha criados no site do Scratch. Na barra de menu do Scratch *desktop,* vá em Arquivo e *click* em "Compartilhar no site". Após isso, aparecerá a aba para dar o nome do projeto, *login* e senha.

Figura 93 – Cadastro no site do projeto Scratch

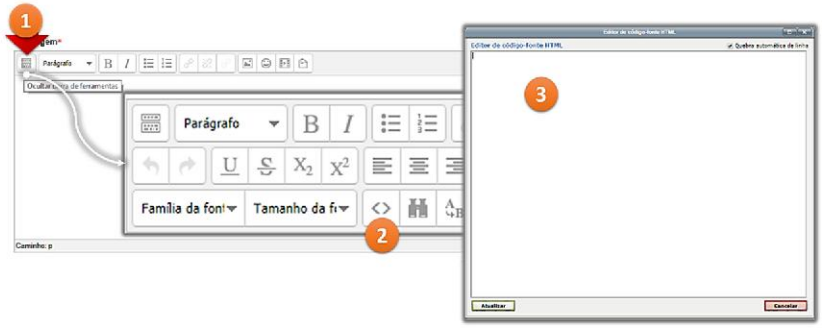
Agora vamos conhecer o "pulo do gato" para incorporar o nosso projeto em uma página com suporte a *flash* no formato html. Para incorporar o projeto já publicado, é necessário copiar o código (destaque da seta na Figura 95).

Figura 94 – Código de incorporação do projeto



Com esse código, vamos colar em páginas com suporte para a linguagem html. Como exemplo, vamos utilizar com o nosso Moodle; acione o botão para ocultar e aparecer a barra de ferramenta (passo 1) e *click* no símbolo "< >" (passo 2). Nessa sequência, vai aparecer a janela para o código de html, é só colar o código copiado no *site* do Scratch na área apresentada pelo passo (3). Não se esqueça de salvar.

Figura 95 – Incorporar no site do Moodle.



Publicar o projeto Scratch no Moodle

A atividade desse tópico é publicar aqui no Moodle o projeto que estamos desenvolvendo.

Vídeo do meu projeto

Esse é o nosso último vídeo, faz parte do meu projeto. Não juntei no módulo anterior devido a questão de recurso computacional, pois um equipamento mais lento poderia ter dificuldades.

https://youtu.be/5PXcjWslLf4

10º Tópico: Pedagoware

A finalidade dessa etapa é compartilhar a relação da tecnologia e a educação.

Pedagogia e Tecnologia Digital

Porque usar tecnologia na educação? Primeiro devemos entender o que é tecnologia. Ao analisar o que é uma tecnologia, encontramos que tecnologia é o estudo de uma técnica que aplica um processo, método, meios e instrumentos dominados pelo ser humano. Então, temos que o alfabeto é uma tecnologia, o quadro negro, o giz..., assim, ao deixar o recado no quadro escrito a giz "a aula será ministrada na sala B", estamos usando uma tecnologia que produziu uma informação e comunicação para quem ler. Ou seja, sempre usamos a Tecnologia da Informação e Comunicação (TIC) para lecionar e nos comunicar com os alunos. Mas as Tecnologias Digitais (TD) disseminam e mudam a forma de aprendermos e nos comunicarmos. Para o ambiente educacional, poderíamos usar o termo de Novas Tecnologias da Informação e Comunicação (NTIC). Todavia, existem duas questões: a primeira é que quando mencionamos tecnologias também temos que são várias e as mais diversificadas, a segunda e mais importante, em minha opinião, pois são questões sobre o que é uma "nova tecnologia" e até quando essa tecnologia é "nova"? Geralmente dizemos a palavra "nova", quando conhecemos, mas, isso não indica que seja nova para a comunidade. Por isso, para o que estamos desenvolvendo e nesse formato de disseminação de conhecimento "sem distância" ou "sem fronteiras" (conhecido como EaD), prefiro o termo Tecnologias Digitais da Informação e Comunicação (TDIC).

Em geral olhamos o termo TIC para área de educação, e é traduzido como uma disciplina, "Introdução à Informática", que quando ministrada por um docente licenciado, se atém a texto e quando ministrado por um docente bacharel da área de sistemas, enfatiza as ferramentas computacionais sem "recorte" pedagógico.

Outra questão sobre as TDICs é o investimento dos órgãos públicos gestores da área de educação.

Sobre a questão de como e onde investir em tecnologia na educação o pesquisador Francislê Neri de Souza, da faculdade de Aveiro, Portugal, em entrevista cedida a Tissianna Carvalhedo, afirma que o processo de aquisição de um *tablet* é mais simples e rápido do que formar o professor para utilizar o recurso tecnológico de forma pedagógica. Para o pesquisador português, o uso da tecnologia (*tablet*) pode ser considerado como um cavalo de Tróia, que mais pode incomodar do que auxiliar na didática da sala de aula (CARVALHEDO, 2016).

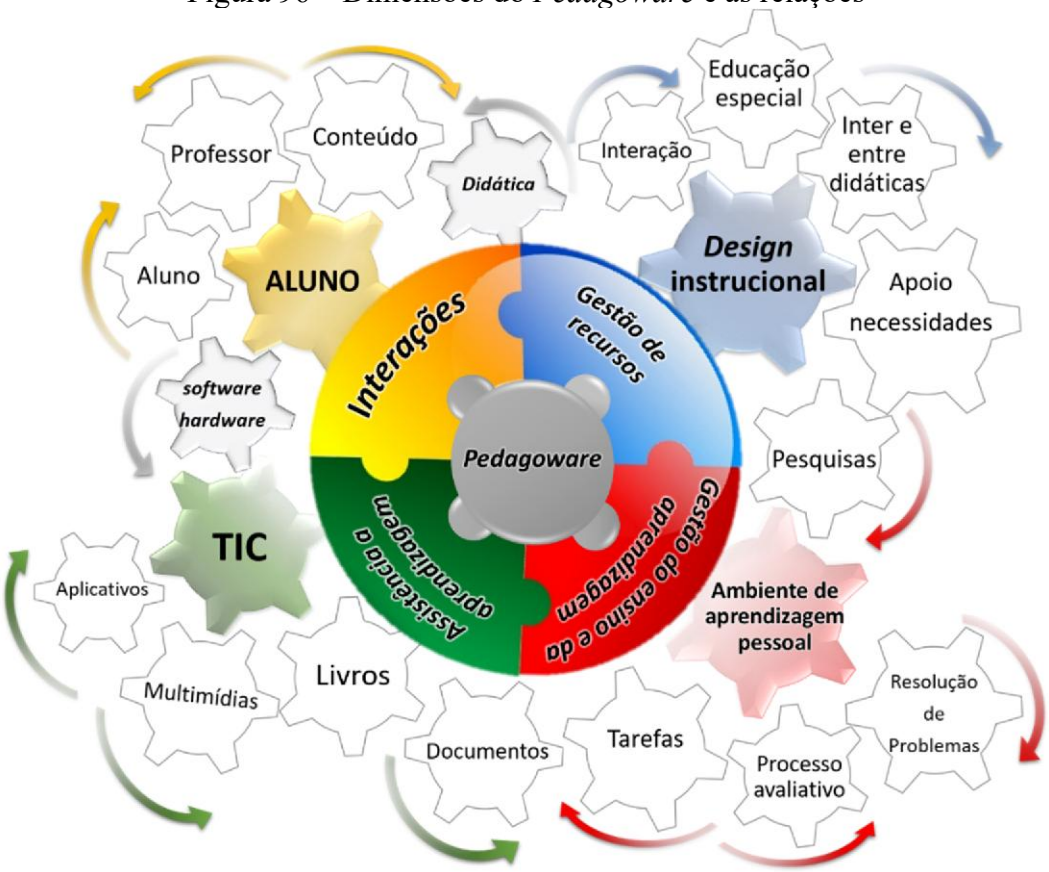
Como qualquer ferramenta, a TDIC pode ser mal empregada. O emprego da tecnologia pode ser superestimado, achando que por si só a tecnologia resolve o problema pedagógico, ou subestimado, quando não se explora todo o potencial da ferramenta, utilizando-se apenas os recursos básicos. Assim, quando é decidido que um docente sem formação em tecnologia e bacharel sem conhecimento pedagógico pode auxiliar no processo de formação do licenciando, estamos definindo um processo caótico para o uso pedagógico da TDIC. Acredito fielmente que devemos pensar em utilizar a tecnologia no ensino dentro da pedagogia, não apenas como

um recurso pronto. Retornando ao pesquisador português, o que vem pronto não se sabe o que é e como está, ou seja, é um cavalo de Tróia, que poderá nos detonar.

O desenvolvimento de material didático, "seja qual for o formato, é concebido refletindo a concepção de aprendizagem na qual seus autores acreditam" (SOUZA; MOL, 2013). Os Autores relatam que o conhecimento do *hardware* e do *software* é importante, entretanto, da mesma forma é necessária a incorporação da pedagogia. Assim, a existência de um terceiro componente é de grande importância para a digitalização da didática, esse componente é o "*pedagoware*". O termo *pedagoware* atua como importante "integração dos elementos: *hardware*, *software*, conteúdo educacional, professor e aluno" (SOUZA; MOL, 2013). É necessário identificar esse conjunto de estratégias, para que atendam a didática, a pedagogia na sua complexidade e as instruções para o ensino e aprendizagem com a TDIC.

Souza e Mol (2013) apresentam o *pedagoware* em quatro dimensões: Gestão de recursos (design instrucional, ou métodos e técnicas pedagógicas; a interação entre os conteúdos e processo didáticos; e necessidades educativas ou especiais); Gestão de ensino e da aprendizagem (ambiente propício para que o aluno desenvolva seu aprendizado); Assistência a aprendizagem (utilização dos recursos das TIC na Educação – livros, aplicativos, multimídias); e Interações (aluno – aluno, aluno – professor, aluno – conteúdo, conteúdo – professor). Apesar da descrição dos autores se completarem ainda e possível identificar que nesse processo incluísse a didática, o *software* e o *hardware* (Figura 97).

Figura 96 – Dimensões do *Pedagoware* e as relações



Fonte: Adaptado de Souza e Mol (2013).

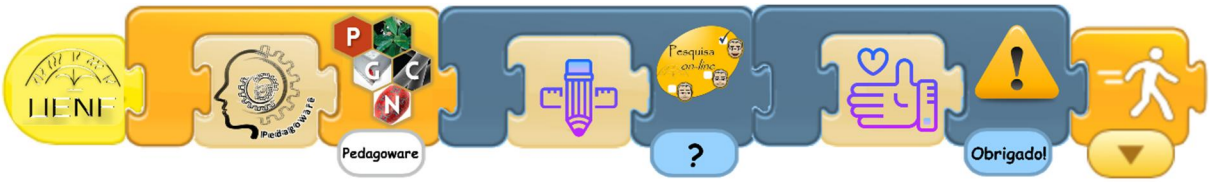
No processo pedagoware, as ferramentas tecnológicas são exploradas de forma crítica e reflexiva para o ensino. Outro ponto a ser observado é a forma de planejamento no projeto pedagógico, pois é comum o emprego de tecnologia em sala de aula, todavia como apoio ao processo pedagógico e recurso potencializador do ensino e aprendizagem.

Porque também gosto desse termo e o adoto? O sufixo "~ware" substantiva questões relacionadas a uma determinada finalidade ou utilidade, ou seja, "coisifica". Isso mesmo, torna um ato abstrato em concreto. E ainda, como área tecnológica, emprega o sufixo ~ware para os principais recursos (*hardware, software, malware, middleware*...), acredito que planejar com recurso da TDIC é pensar de como promover a aprendizagem com "~ware".

Contudo, penso em pedagoware como planejamento do uso da tecnologia, onde deve haver um projeto, esse projeto deve ser bem pensado e sempre que possível discutido, testado e somente depois implementado em sala. A implementação advém de um formato pedagógico para proporcionar a aprendizagem e não só o ensino, pois penso que o ensino parte do docente e a aprendizagem é uma ação cooperativa entre quem fornece e recebe a informação.

Figura 97 – O Pedagoware a TDIC e Aprendizagem

Eu observo pedagoware como uma engrenagem pequena e simples que conecta o uso da tecnologia (TDIC) e da pedagogia para proporcionar a aprendizagem. E para isso, vejo a necessidade de conhecer e planejar, assim como aprendemos a lecionar e planejarmos o ensino, para que ocorra a aprendizagem.



Referências utilizadas para montar esse material.

AUSUBEL, D. P. **Aquisição e Retenção de Conhecimentos**: Uma Perspectiva Cognitiva. Tradução: Lígia Teopisto Lisboa - Portugal: Paralelo Editora. 2003. 243 p.

CARVALHEDO, T. **O uso do tablet na educação e o PedagowareTICs e EaD em Foco.** São Luis: [s.d.]. Disponível em: <http://www.uemanet.uema.br/revista/index.php/ticseadfoco/article/view/51/163#>.

FANNING, Ruth M.; GABA, David M.. The Role of Debriefing in Simulation-Based Learning. **Jornal Of The Society Simulation In Healthcare**, Philadelphia, v. 2, n. 2, p.115-125, abr. 2007. Bimestral. Doi: 10.1097/SIH.0b013e3180315539.

FORD JR, Jerry Lee. **Scratch Programming for Teens tm**. Canada: Course Technology, 2009.

KIRNER, Claudio. Introdução à realidade virtual (Mini-Curso). In: WORKSHOP DE REALIDADE VIRTUAL, 1., 1997, São Carlos. **Anais... .** São Carlos: Ufscar, 1997. p. 40.

MARJI, Majed. **Aprenda a programar com Scratch**, 1. Ed. São Paulo: Novatec, 2014.

MOREIRA, Marco Antonio. **A teoria da aprendizagem significativa e sua implementação em sala de aula**. Brasília: Editora Universidade de Brasília, 2006. 186 p.

PANITZ, Theodore**. Collaborative versus Cooperative Learning: A Comparison of the Two Concepts Which Will Help Us Understand the Underlying Nature of Interactive Learning**. Washington - EUA: ERIC, Institute of Education Sciences, 1999, 13p. Disponível em: <http://files.eric.ed.gov/fulltext/ED448443.pdf>. Acesso em 07 jul. 2017

ROCHA, Ana Regina; CAMPOS, Gilda H. Bernadino de. Avaliação da qualidade de software educacional, **Em Aberto**, v. 12, n. 57, 1993. Disponível em: <http://rbep.inep.gov.br/index.php/emaberto/article/viewFile/1879/1850>. Acesso em 2 ago. 2017

RESNICK, Mtchel et al. Scratch: Programing fo all. **Comunications of the ACM**, v. 52, n. 11, p. 60-67, 2009

SCHOSSER, Rejane Leal. A atuação de tutores nos cursos de Educação a Distância, **Revista Digital da CVA – Ricesu**, v. 6, n. 22, 2010.

SOUZA, F. N. de; MOL, G. S. Livro didático digital de química: princípios para a construção

em tabletes. Enseñanza de las ciências: revista de investigación y experiências didáticas. **Anais...** Girona, Espanha: Enseñanza de las ciências, 2013.

TAVARES, Romero et al. Objetos de aprendizagem: uma proposta de avaliação da aprendizagem significativa. In: PRATA, Carmem Lúcia; NASCIMENTO, Anna Christina Aun de Azevedo Orgs). **Objetos de aprendizagem...**. Brasília: MEC, 2007. p. 123-134. Disponível em: <http://rived.mec.gov.br/artigos/livro.pdf>. Acesso em: 24 jun. 2015.

VALENTE, José Armando. A comunicação e a educação baseada no uso das tecnologias digitais de informação e comunicação. **UNIFESO** - Humanas e Sociais. V. 1, n. 01, p. 141-166, 2014. Disponível em: <http://www.revistasunifeso.filoinfo.net/index.php/revistaunifesohumanasesociais/article/view/17/24. Acesso em 21 ago. 2017.

WILEY, David A.. Connecting learning objects to instructional design theory… . In: _____.**The Instructional Use of Learning Objects:**Online Version. e-book: Ait / Aect, 2000. 1. 35. Disponível em: <http://reusability.org/read/chapters/wiley.doc>. Acesso em: 07 jul. 2015.

Glossário

Algoritmo: Sequência finita de instruções racionais, com roteiros lógicos sem ambiguidades.

Ambiente de programação: pode ser pensado em interface que permite unir uma área de interação do homem com a máquina. No caso específico para a programação, permite criar bytecodes (código em bytes).

Argumento: são valores atribuídos a um parâmetro. Ou seja, em Scratch estamos pensando em processos que acionam o bloco.

Banda larga: conexão com a internet de velocidades superiores a 56k (conexão discada). No contexto de banda larga, a conexão permite visualizar vídeos, áudios e videoconferências; é executada *on-line,* com desempenhos aceitáveis de estabilidade, geralmente acima de 256Kbts.

Booleano: em computação, refere-se ao formato de dado, que pode assumir dois valores: verdadeiro ou falso. Esse valor pode ser representado por "0" ou "1". A expressão booleana está relacionada com a álgebra de George Boole, onde é considerado "0" como falso e "1" como verdadeiro. Ou seja, "1" existe alguma informação e "0" não existe informação.

Compilar (compilação): refere-se a processo de conversão do script em uma linguagem que permite a execução fora do ambiente de origem, utilizando um programa tradutor. A definição linguística de compilar é unir, juntar ou reunir; esse significado faz sentido na computação quando pensamos que o ato de compilar é reunir todos os códigos da linguagem em um objeto, para que o sistema computacional interprete.

Delay: termo de origem do idioma inglês, que define o retardo de sinais em relação ao tempo de execução.

Feedbacks: termo de origem do idioma inglês, que define a reação do receptor, ou realimentação de uma informação enviada.

Interface: é o *layout* ou elemento que apresenta a ligação entre dois ambientes, ou partes de sistemas que não se conectam diretamente.

Interpretar (interpretação): a interpretação é a execução do código computacional (*script*), sem necessitar de um código executável gerado pelo sistema de "compilação". Sistemas como Scratch, Java, Phyton, etc. Entretanto a interpretação é visual na tela, porém, há uma compilação para que o sistema computacional compreenda.

Logado (*login*): termo de origem do idioma inglês, que define que está inserido em um lugar. Para a área computacional define-se como está autorizado a acessar as informações da base de dado para qual foi identificado.

Login (log in): termo de origem do inglês, significa acesso ao lugar ou dentro.

Logo (*software*): é uma linguagem de programação com interface simplificada para atender ao público infantil.

Logomarca: é a identidade de uma organização, instituição ou pessoa, que tem uma representação gráfica.

Loop: para a área computacional, é o conjunto de instruções, que são repetidas até que uma condição seja satisfeita.

Moodle: plataforma digital modular de ensino e aprendizagem dinâmica orientada a objetos (Moodle - Modular Object-Oriented Dynamic Learning Environment), onde as interações ocorrem por intermédio de ferramentas, como *fórum*, *chat*, imagens, vídeos...

Navegador ou *browser*: é a interface utilizada para acessar as "*webpages",* que permite a navegação na web.

Networkin: é um termo do inglês, que representa uma rede de contatos de um indivíduo.

Parâmetro: em ciência de computação é utilizado como argumento, mas para programação refere-se ao dado personalizado de um procedimento, de uma variável.

Plug-in: termo de origem do idioma inglês que significa adicionar funções a um determinado *software*, geralmente é aplicado a navegadores para internet.

QRCode (Quick Response Code): Código bidimensional do código de barra. A tradução significa Código de resposta rápida.

Resetar (resetando, resetado - *reset*): termo de origem do idioma inglês, que significa voltar para a configuração inicial.

Scratcher: é a pessoa que usa o Scratch

Script: roteiros para serem seguidos. Na computação, refere-se às instruções por códigos que controlam um determinado evento, com sequências que visam atender um objetivo.

Semântica: em linguística, significa a interpretação de uma palavra, em computação, a semântica é a forma de escrever a os programas e as estruturas de programação.

Setado: em computação, significa que o valor foi informado e aceito.

Sintaxe: regras que regem a formulação textual. Na área computacional, significa o conjunto de estruturas que um determinado software decodifica.

Script: conjunto de informações a serem lidas e executadas pelo sistema computacional.

Spriter: palavra de origem do latim, que quer dizer espírito significando duende, fada ou elfo. Na computação, refere-se a um elemento gráfico que pode deslocar-se na tela sem que haja marcas, como uma pequena criatura com poderes mágicos.

String: é uma sequência ou cadeia de caracteres alfanuméricos. No Scratch, é possível comparar e saber qual caractere está em determinada posição da *string*. É considerado qualquer caractere, incluindo espaço e ponto, como está representado abaixo a *string* "SCRATCH ver2.0".

String	**S**	**C**	**R**	**A**	**T**	**C**	**H**		**V**	**e**	**r**	**2**	**.**	**0**
Posição	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*

Tag, termo em inglês que significa etiqueta. Na área computacional, significa que são estruturas de linguagem de marcação, contendo instruções. Cada ambiente possui uma tag diferente.

Webcam: câmara de vídeo aplicada ao recurso de desktop, geralmente de baixa performance.