O3 – Conteúdo de apoio instrucional

Colectânea de fichas de aprendizagem através de game-design dirigida a professores

Dados do documento

Documento: O3/A1 - Coletânea de fichas de aprendizagem de jogos dirigida a professores

***Intellectual Output* Nº - Título**: O3 – Conteúdo de apoio instrucional

Coordenador do *Intellectual Output*: Universidade South-West "Neofit Rilski" (Bulgária)

Parceiros envolvidos: Universidade de Liubliana (Eslovénia), Universidade de Rijeka (Croácia)

Isenção de responsabilidade

O apoio da Comissão Europeia à produção desta publicação não constitui um aval do seu conteúdo, que reflete unicamente o ponto de vista dos autores, e a Comissão não pode ser considerada responsável por eventuais utilizações que possam ser feitas com as informações nela contidas.

FICHAS DE APRENDIZAGEM

As fichas de aprendizagem preparadas apresentam as atividades, desde as mais básicas, com apenas um conceito de programação, até às mais avançadas, que englobam vários conceitos. A tabela que se segue mostra a ordem proposta para a realização das atividades.

CENÁRIOS DE APRENDIZAGEM BÁSICOS		
1	**Introdução à plataforma do Snap!** Conhecer a plataforma de programação do Snap!	UL
2	**É o momento de dar vida ao teu *sprite*** Descobrir blocos de código, conectá-los, mover o *sprite* e fazê-lo dizer algo	UL
3	**Mover-se pelo *stage*** Criar uma sequência de blocos significativa	UL
4	**Mudar de traje e virar**	UL
5	**Sons da quinta** Adicionar, importar, gravar e reproduzir som	UL
6	**As férias de verão do camaleão, versão simples** Perceber o que são eventos, cores, valores Booleanos, verificar e responder a dois estados de jogo diferentes	UL
7	**Ajudar o Príncipe e a Princesa a encontrarem os seus animais** Usar condicionais, desenhar	UL
8	**Desenhar com giz** Usar *loops*, virar, alterar o fundo	UL
9	**Apanhar o lixo e limpar o parque** Trabalhar com variáveis, duplicar *sprites*, blocos de código	UL
10	**Alimentar os gatos** Usar variáveis (dentro/fora do *loop*), *loops*, números aleatórios, concatenar *strings*, operadores, controlos	UL
11	**Adivinhar o número de gatos no abrigo** Usar valores aleatórios, input de variáveis, condicionais, operadores de comparação, contadores	UL
CENÁRIOS DE APRENDIZAGEM AVANÇADOS		
12	**Apanhar comida saudável** Uso de variáveis, condicionais, *loops*, pontos de direção, valores aleatórios	UL
13	**Storytelling**	SWU
14	**Desenhar**	UNIRI

15	**Apanhar o rato** Usar *loops*, condicionais, variáveis	UL
16	**Comprar comida para um piquenique** Usar variáveis, condicionais, operadores	UL
17	**Operações**	SWU
18	**Reciclar**	SWU
19.1	**Tocar piano 1**	SWU
19.2	**Tocar piano 2**	UNIRI
20	**Teste**	SWU
21	**Jogo simplificado do PACMAN** Usar movimento do objeto baseado em eventos, sensores de cores, valores Booleanos, verificar e responder a dois estados de jogo diferentes	UL

INTRODUÇÃO

Ao longo do último século, importantes especialistas da área da psicologia identificaram os jogos como atividades cruciais para o desenvolvimento de competências muito importantes na criança, independentemente da faixa etária ou fase de desenvolvimento. Através do jogo, esta pode aprender a adaptar-se facilmente a novas circunstâncias e a lidar melhor com a mudança. Ao brincar, pode descobrir conceitos básicos sobre o mundo real e estabelecer as primeiras ligações entre eles.

Atualmente, os jogos são utilizados principalmente nas primeiras fases de desenvolvimento da criança, em casa ou no infantário. A aprendizagem na escola ainda é extremamente baseada num modelo tradicional em que o professor transmite o conhecimento e os alunos ocupam uma posição passiva. Por outro lado, as teorias de aprendizagem desenvolvidas ao longo do último século promovem novas abordagens de educação e aprendizagem, mais centradas nos estudantes, baseadas na resolução de problemas, direcionadas para objetivos educacionais superiores, motivadoras e, frequentemente, apoiadas pelas TIC.

A abordagem do CODING4GIRLS irá encorajar a participação em atividades de programação, através de uma "*low entry high ceiling approach*", que irá requerer um conhecimento baixo no início, sem, ainda assim, limitar os desafios de resolução de problemas para alunos mais avançados. Os alunos irão ser estimulados a terminar problemas parcialmente resolvidos ao acrescentarem blocos de código em falta ou ao criarem as suas próprias soluções. As atividades estão planeadas sequencialmente - umas são mais básicas, com apenas um conceito de programação, e outras são mais complexas, contendo vários conceitos. Ao desenvolvermos as atividades de aprendizagem no Snap!, focámo-nos nas características identificadas dos jogos preferidos pelas raparigas e nas atividades relacionadas com problemas do mundo real.

As fichas de aprendizagem preparadas apresentam, de forma concisa, informação que irá ajudar os instrutores a integrar os jogos sérios propostos e as metodologias de *design thinking* nas suas práticas de ensino. As fichas seguem o design das metodologias do CODING4GIRLS e incluem informação para a preparação das atividades destinadas a desenvolver as capacidades de programação de rapazes e raparigas. A informação disponibilizada é a seguinte:

- Objetivo geral de aprendizagem para cada atividade
- Conceitos abrangidos pela atividade
- Objetivos de aprendizagem específicos
- Resultados de aprendizagem esperados
- Guia passo-a-passo para a utilização da metodologia de aprendizagem CODING4GIRLS
- Métodos de avaliação do conhecimento desenvolvido
- Questões de partida para discussões entre os alunos

Foram desenvolvidas 21 fichas de aprendizagem, uma para cada uma das atividades preparadas. Os professores podem usar os cenários e os jogos seguindo a ordem proposta ou podem optar por outra, de acordo com as suas preferências e necessidades. As fichas incluem a função genérica do jogo sério proposto, incluindo os processos de interação do utilizador e o feedback, assim como descrições de todas as atividades de aprendizagem que serão implementadas no jogo sério.

As fichas de aprendizagem estão disponíveis em Inglês, assim como nos idiomas nacionais dos parceiros do projeto - Búlgaro, Croata, Grego, Italiano, Português, Esloveno e Turco.

FICHAS DE APRENDIZAGEM

As fichas de aprendizagem preparadas apresentam as atividades, desde as mais básicas, com apenas um conceito de programação, até às mais avançadas, que englobam vários conceitos. A tabela que se segue mostra a ordem proposta para a realização das atividades.

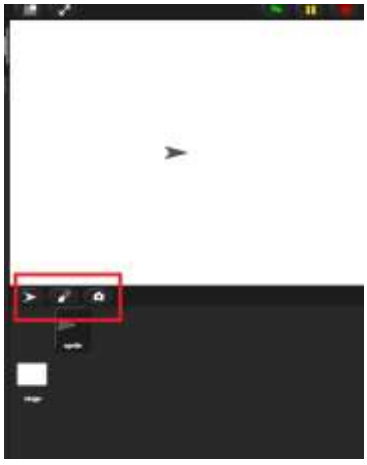
CENÁRIOS DE APRENDIZAGEM BÁSICOS		
1	**Introdução à plataforma do Snap!** Conhecer a plataforma de programação do Snap!	UL
2	**É o momento de dar vida ao teu *sprite*** Descobrir blocos de código, conectá-los, mover o *sprite* e fazê-lo dizer algo	UL
3	**Mover-se pelo *stage*** Criar uma sequência de blocos significativa	UL
4	**Mudar de traje e virar**	UL
5	**Sons da quinta** Adicionar, importar, gravar e reproduzir som	UL
6	**As férias de verão do camaleão, versão simples** Perceber o que são eventos, cores, valores Booleanos, verificar e responder a dois estados de jogo diferentes	UL
7	**Ajudar o Príncipe e a Princesa a encontrarem os seus animais** Usar condicionais, desenhar	UL
8	**Desenhar com giz** Usar *loops*, virar, alterar o fundo	UL
9	**Apanhar o lixo e limpar o parque** Trabalhar com variáveis, duplicar *sprites*, blocos de código	UL

10	**Alimentar os gatos** Usar variáveis (dentro/fora do *loop*), *loops*, números aleatórios, concatenar *strings*, operadores, controlos	UL
11	**Adivinhar o número de gatos no abrigo** Usar valores aleatórios, input de variáveis, condicionais, operadores de comparação, contadores	UL
CENÁRIOS DE APRENDIZAGEM AVANÇADOS		
12	**Apanhar comida saudável** Uso de variáveis, condicionais, *loops*, pontos de direção, valores aleatórios	UL
13	**Storytelling**	SWU
14	**Desenhar**	UNIRI
15	**Apanhar o rato** Usar *loops*, condicionais, variáveis	UL
16	**Comprar comida para um piquenique** Usar variáveis, condicionais, operadores	UL
17	**Operações**	SWU
18	**Reciclar**	SWU
19.1	**Tocar piano 1**	SWU
19.2	**Tocar piano 2**	UNIRI
20	**Teste**	SWU
21	**Jogo simplificado do PACMAN** Usar movimento do objeto baseado em eventos, sensores de cores, valores Booleanos, verificar e responder a dois estados de jogo diferentes	UL

CENÁRIOS DE APRENDIZAGEM BÁSICOS

Cenário de aprendizagem 1 - Introdução à plataforma do Snap!

Título de cenário de aprendizagem	Introdução à plataforma do Snap!
Experiência prévia de programação	/
Objetivos de aprendizagem	Objetivos de aprendizagem gerais: ● conhecer a plataforma de programação Snap! Objetivos de aprendizagem específicos: ● O estudante é capaz de adicionar um novo *sprite* ● O estudante é capaz de adicionar um traje a um *sprite* e editá-lo ● O estudante é capaz de centrar o *sprite*, de forma a que a rotação funcione adequadamente ● O estudante é capaz de adicionar um novo fundo ao *stage* e editá-lo
Objetivo, tarefa e breve descrição das atividades	Os estudantes adicionam um *sprite*, um traje a esse *sprite*, editam-no, e apagam um deles. O estudante cria um fundo para o *stage*, edita-o, e apaga os indesejados. Objetivo: Ao fim de uma hora, os estudantes irão desenhar a sua personagem preferida e o seu ambiente, real ou imaginário, para utilizar no contexto de um jogo. De forma a tornar a atividade mais motivadora para todos os estudantes, o desenho de *sprites* foi identificado em estudos científicos como sendo adequado para este grupo-alvo.
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Demonstração pelo professor Trabalho individual
Formas de ensino	Trabalho presencial Trabalho individual

Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) Ao fim de uma hora, os estudantes irão desenhar a sua personagem preferida e o seu ambiente, real ou imaginário, para utilizar no jogo. [Passo 1] Mostre aos estudantes a página onde podem encontrar o Snap! (https://snap.berkeley.edu/). Mostre-lhes as várias partes da plataforma: a secção com os blocos, a secção onde podem montar guiões/mudar trajes/adicionar sons, um *stage* com o *sprite*, a lista dos *sprites*.  [Passo 2] Podes criar um *sprite* clicando num dos três botões seguintes: 

Irás desenhar um novo *sprite*, por isso seleciona o pincel, e uma nova janela irá aparecer, onde poderás desenhar o *sprite* da mesma forma como farias no Paint.
Tarefa para os estudantes: Desenha o teu primeiro *sprite*. Tens 10 minutos.
Depois de o *sprite* estar desenhado, deves garantir que o centro de rotação do *sprite* está onde queres. Para tal, usa  .
Tarefa para estudantes: centra o teu *sprite*.

[Passo 3]
Para editar o teu *sprite*, escolhe o separador Trajes, que apenas será visível quando o teu *sprite* estiver a ser clicado. Clica com o lado direito do rato num traje que queiras editar, e seleciona "editar". No mesmo menu, podes também duplicar ou apagar o teu traje.

[Passo 4]
Para importar um traje já existente, clica no ícone com uma folha de papel, e escolhe Trajes...

Mais uma vez, esta opção só irá aparecer quando o teu *sprite* estiver a ser selecionado no *stage*.
Tarefa para estudantes: seleciona um traje e adiciona-a ao teu *sprite*.

[Passo 5]

Agora já tens a tua personagem; é altura de colocares um fundo no *stage*. Para tal, clica no *Stage*, em vez de na personagem. Para adicionar um novo fundo, seleciona o separador Backgrounds.

Tarefa para estudantes: desenhem o vosso próprio cenário.
Tarefa para estudantes: procura entre os cenários existentes e importa um, de forma a ficares com dois.
Tarefa para estudantes: Encontra uma forma de editares o teu cenário. Depois, descobre como apagar um dos dois, de forma a ficares com um só.

	Reflexão e avaliação: Foram os estudantes capazes de desenhar a sua personagem e o ambiente onde esta vive? Tiveram problemas a fazê-lo? Como os resolveram?
Instrumentos e recursos para o Professor	https://snap.berkeley.edu/
Recursos/Materiais para os alunos	Instruções para o estudante (C4G1_InstructionsForStudent.docx)

Cenário de aprendizagem 2 - É o momento de dar vida ao teu *sprite*

Título do cenário de aprendizagem	É o momento de dar vida ao teu sprite
Experiência prévia de programação	/
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● O estudante sabe onde encontrar blocos de código e como conectá-los para formarem uma sequência ● O estudante sabe como movimentar o seu *sprite* ● O estudante sabe como pôr o *sprite* a dizer algo Objetivos de aprendizagem específicos orientados para o pensamento algorítmico: ● Fazer uma sequência de blocos significativa
Objetivo, tarefa e breve descrição das atividades	Os estudantes descobrem onde estão guardados os blocos de código e como escolher os mais apropriados, que categorias de blocos existem, e como os conectar para formarem sequências.
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Demonstração pelo professor Trabalho individual
Formas de Ensino	Ensino presencial Trabalho individual
Sumário da lição	((Motivação-Introdução, Implementação, Reflexão e Avaliação) Durante esta hora, irás aprender a movimentar a tua personagem, assim como fazê-la falar. Os professores podem mostrar aos estudantes um exemplo de algo que irão programar nesta hora. [Passo 1] Em primeiro lugar, vamos olhar para os blocos de código que estão disponíveis para utilizares. Onde estão eles? Do lado esquerdo, consegues encontrar diferentes categorias de blocos: Movimento, Aparência, Caneta, Controlo, Sensores, Operadores e Variáveis. Para começar, vamos utilizar os blocos  .

Tarefa para alunos: Encontra o bloco e clica duas vezes nele. O que aconteceu?

[Passo 2]

Para começar a conectar os blocos, tens de arrastar e largar os teus blocos 

até ao separador Guiões.

Podes clicar duas vezes no bloco dentro do separador Guiões para executar o código.

[Passo 3]
Os programas no Snap! são iniciados, normalmente, clicando na bandeira verde.
Tarefa para estudantes: vai clicando nas várias categorias e tenta encontrar um bloco que inicie o programa ao clicar na bandeira verde.

Solução:

Se quiseres que um programa funcione numa sequência de passos correta, os blocos têm de estar conectados, tal como os puzzles. Assim:

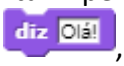
Agora, de cada vez que clicares na bandeira verde, o *sprite* vai mover-se 10 passos, a partir de posições diferentes na imagem.

[Passo 4]

Se um bloco tiver um espaço branco, significa que podes alterar os números ou letras escritas.
Tarefa para estudantes: Faz com que a tua personagem se mova 30 passos de cada vez, e não 10.

[Passo 5]

Faz com que a tua personagem diga algo. Onde podes encontrar o bloco "diz"? Tenta perceber a diferença entre 

e 

, e explica-a ao teu colega.

[Passo 6]

Encontraste ambos os comandos na categoria Aparência. A principal diferença é que quando selecionas 

, não dizes ao programa para esperar ____ segundos antes de o código continuar, ou que deve parar de falar a determinada altura.

[Passo 7]

Seleciona a tua personagem. Ao arrastá-la para o *stage*, move-a para o lado esquerdo e programa um código que a faça movimentar-se 

da sua posição à esquerda para o lado direito do *stage*. Depois de cada movimentação, a personagem deve dizer algo.

Move-a mais do que uma vez.

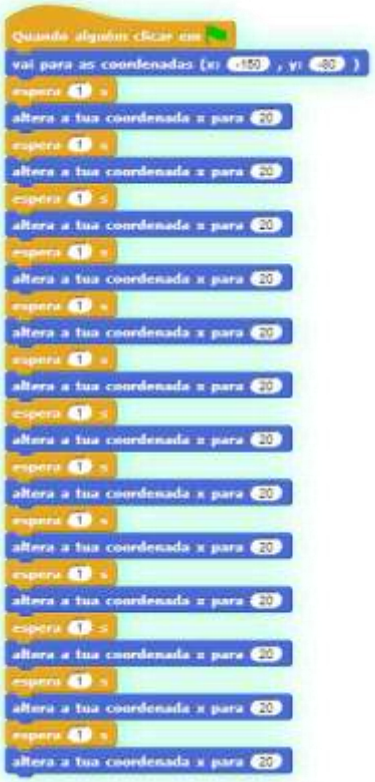
Experimenta. A personagem ficou exatamente na mesma posição de cada vez que executaste o programa?

Consegues encontrar um bloco que garanta que a tua personagem comece sempre na mesma posição e não saia do *stage*?

	Dica para o professor: se a personagem sair do *stage*, pode recuperá-la clicando nele com o lado direito do rato e escolhendo "*mostrar*". O bloco que procuras é  . Para determinar quais "x" e "y" estão certos, podes mover a tua personagem para o local onde queres que ela esteja, clicar em "a coordenada x da posição" e a "a coordenada y da posição" (no fim da categoria Movimento) e irás conseguir ver o "x" e o "y" atuais. Tens de escrever esses números nos espaços em branco e ir para o bloco. Reflexão e avaliação: Quantas vezes teve a tua personagem de completar a sequência de se movimentar e falar? Foi o mesmo Número de vezes que o resto da turma? Qual é a razão para tal?
Instrumentos e recursos para o Professor	**Exemplo de programa:** https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home
Recursos/Materiais para os alunos	• Instruções para alunos (C4G2_InstructionsForStudent.docx) • Se o estudante não tiver desenhado o seu *sprite* e cenário, pode utilizar: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home_tmp

Cenário de aprendizagem 3 – Mover-se pelo *stage*

Título do cenário de aprendizagem	Mover-se pelo *stage*
Experiência prévia de programação	● O estudante sabe onde encontrar blocos de código e como os conectar de forma a construir uma sequência.
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Criar uma sequência significativa de blocos Objetivos de aprendizagem específicos, orientados para o pensamento algorítmico: ● O estudante posiciona o *sprite* no *stage*. ● O estudante altera a posição x e y do *sprite* ● O estudante repete X vezes o *loop* ● O estudante sabe que a direção de movimento do *sprite* em ___ passos é relativa à direção para a qual o *sprite* está orientado
Objetivo, tarefa e breve descrição das atividades	Breve descrição: O estudante aprende a movimentar o seu *sprite* na direção x e y, programa para resolver as tarefas propostas, aprende a virar o seu *sprite* para uma direção diferente e percebe como tal afeta o bloco “mover _____ passos”. Tarefas: criar um programa que movimente o *sprite* na direção e y, criar um programa que combine movimento e as direções x e y. Objetivo: diferenciar entre movimento nas direções x e y e repetir o *loop.*
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Demonstração pelo professor Trabalho individual
Formas de ensino	Ensino presencial Trabalho individual

Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e Avaliação) Irás ajudar diferentes animais a cumprirem os seus objetivos. Para tal, irás precisar de lhes dar instruções sobre como se movimentarem pelo *stage*. [Tarefa 1] Abre Catch *the ball* e adiciona código de forma a que o cão apanhe a bola. Usa os blocos  e  para fazer uma animação de um cão a movimentar-se em direção à bola. Uma possível solução para esta tarefa seria:  Como podes ver, o x muda quando te moves para a direita ou para a esquerda. Se o x for 0, o teu *sprite* está no meio do *stage*. Tudo o que está à esquerda, quanto mais à esquerda estiver, maior é o seu número. À direita do meio, os valores de x são maiores que 0.

Dica: Se a atividade for feita com estudantes mais velhos, que já conhecem casas decimais, o tempo de espera pode ser menor, por exemplo 0.1. Se eles já souberem o que é um sistema de coordenadas, algumas explicações podem ser omitidas.

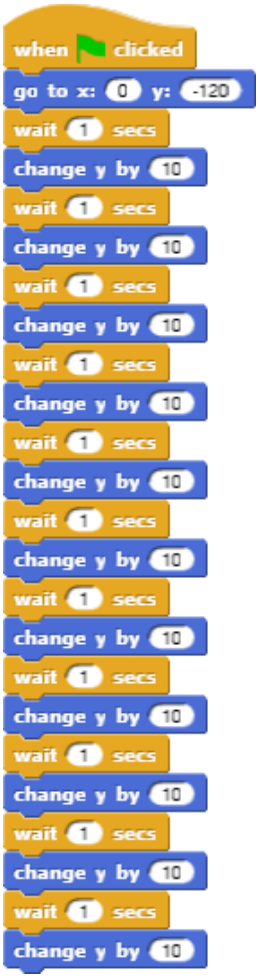
[Tarefa 2]

Abre *Help monkey climb the tree* e adiciona um código ao macaco para que ele apanhe as bananas. Usa os blocos 

e 

para fazer uma animação do macaco a trepar a palmeira.

Uma possível solução para esta tarefa seria:



Como podes ver, o y muda quando te moves para cima ou para baixo.

Se o y for 0, o teu *sprite* está no meio do *stage*. Tudo o que estiver mais alto do que o meio tem um x superior a 0. Se quiseres que o teu *sprite* esteja abaixo da linha média do *stage*, pensa que é como se estivesses a mergulhar – a profundidade a que estás depende do número de metros que colocas depois do "- ". Aqui, esse número corresponde ao número de passos abaixo da linha do meio a que estás. Se quiseres descer da árvore, usa

.

Dica: Se a atividade for feita com estudantes mais velhos, que já conhecem casas decimais, o tempo de espera pode ser menor, por exemplo 0.1. Se eles já souberem o que é um sistema de coordenadas, algumas explicações podem ser omitidas.

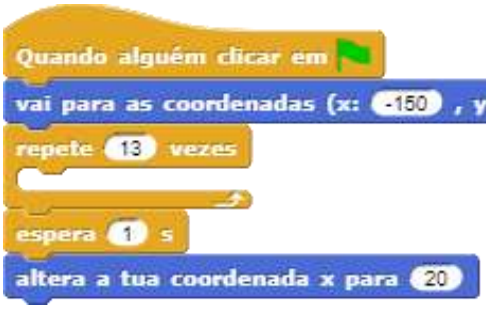
[Passo 3]

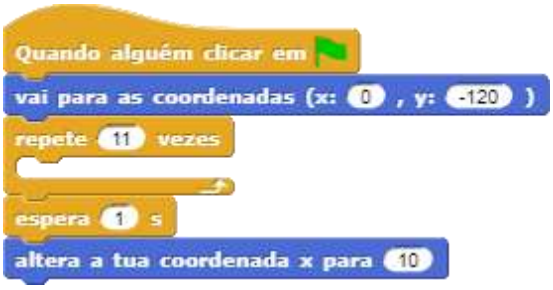
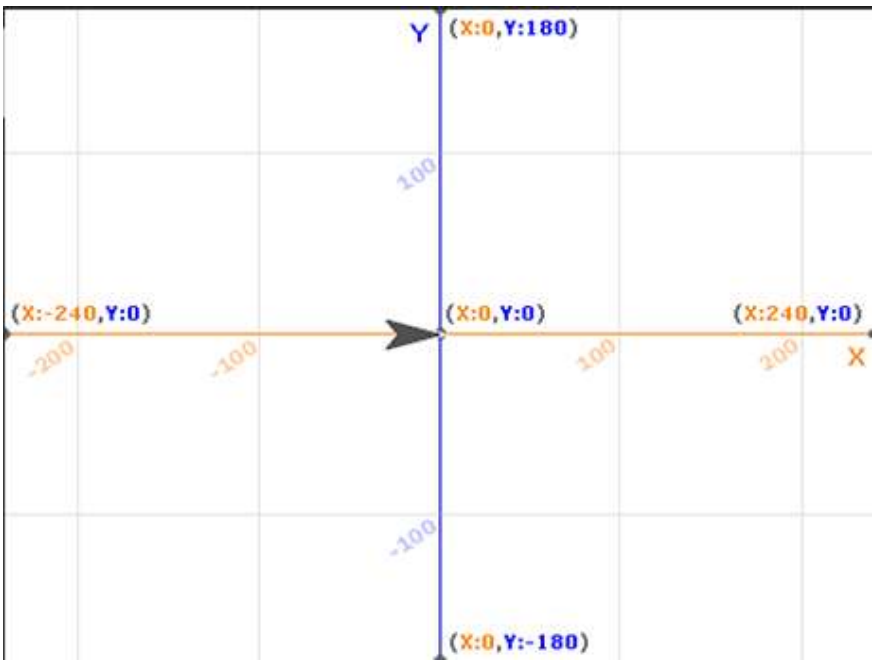
Tiveste, em ambas tarefas, de forma intercambiável, usar dois blocos. Quantas vezes tiveste de **repetir o código**?

Há uma forma mais rápida de escrever este código, ordenando ao teu computador que o repita um certo número de vezes. Este é o *loop* "repete ____". Podes utilizá-lo quando uma mesma ação ou sequências de ações se repetem mais do que uma vez. Tenta repetir o código para ambas as tarefas, usando o *loop*

. O código que queres repetir tem de ser colocado dentro deste bloco, e tens de escrever no espaço em branco quantas vezes deve ser repetido.

Código para o cão:

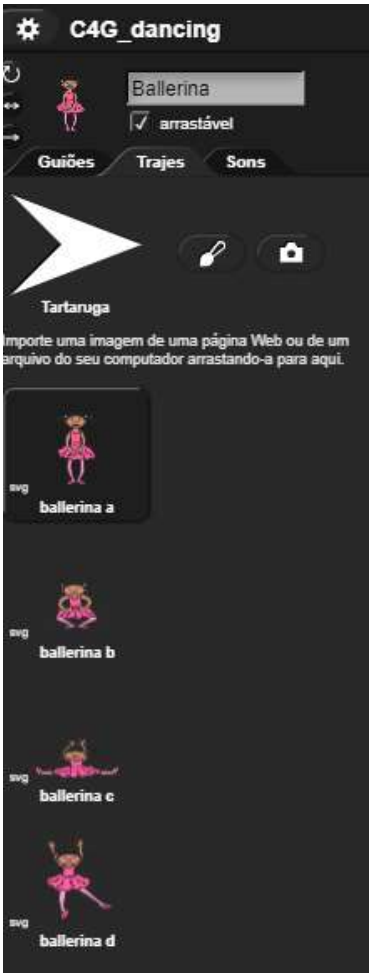


	Código para o macaco:  Tarefa: Tenta fazer com que o cão corra até à bola e volte. Tarefa: Tenta fazer com que o macaco suba até ao topo da árvore e desça. Do que gostaste mais? Para ser mais fácil veres a posição do x e do y no *stage*, podes usar o cenário da grelha XY no Snap: 
Instrumentos e recursos para o Professor	• Uma possível solução para apanhar a bola: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_x

	● Uma possível solução para ajudar o macaco a trepar à árvore: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_y
Recursos/Materiais para os alunos	• Apanha a bola: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Catch_the_ball ● Ajuda o macaco a subir à árvore: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Help_monkey_climb_the_tree ● Instruções para o estudante (C4G3_InstructionsForStudent.docx)

Cenário de aprendizagem 4 – Mudar de traje e virar

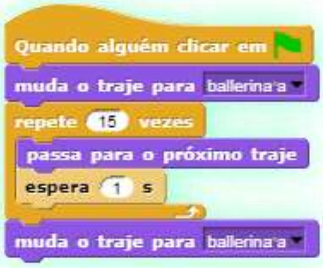
Título do cenário de aprendizagem	Mudar de traje e virar
Experiência prévia de programação	Movimento
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Construir uma sequência significativa de blocos Objetivos de aprendizagem específicos, orientados para o pensamento algorítmico: ● O estudante muda o traje do *sprite* para fazer uma animação ● O estudante muda a rotação das personagens
Objetivo, tarefa e breve descrição das atividades	**Breve descrição:** O estudante aprende a mudar o traje para fazer uma animação. Aprende também como alternar entre vários tipos diferentes de rotação do *sprite*. **Tarefas:** criar um programa que mude o traje do *sprite* e selecionar, para cada programa, o tipo apropriado de rotação para cada *sprite*. **Objetivo:** saber como alterar o traje do *sprite* e como escolher o tipo adequado de rotação para ele.
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Demonstração por parte do professor Trabalho individual
Formas de ensino	Ensino presencial Trabalho individual

Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) Irás aprender como animar um *sprite* de forma a que pareça que está a andar, a dançar… [Passo 1] Abre um novo projeto vazio, clica no ícone que parece uma folha de papel, e seleciona Trajes… Clica na bailarina a, e depois em Importar. Faz o mesmo com a bailarina b, c e d. No separador Trajes, tens agora quatro trajes de bailarina. Podes mudar o nome do *sprite* para "bailarina" na barra acima do separador Trajes: 

Agora volta ao separador Guiões e tenta criar um código que seja iniciado quando se clica na bandeira verde, que a cada segundo mude a aparência da bailarina, num total de 15 vezes. Irás precisar de usar o bloco 

. Garante que tua bailarina começa e termina a sua dança com as duas pernas no chão. As posições iniciais e finais não fazem parte da sua dança.

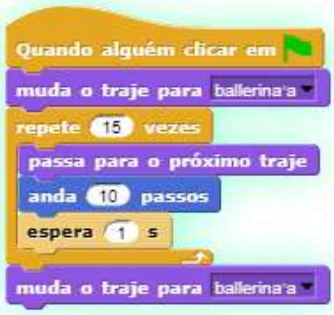
Solução:



[Passo 2]

A nossa bailarina não quer estar na mesma posição o tempo todo, por isso faz pequenos movimentos de cada vez que muda de roupa. Adiciona este movimento à sua dança.

Possível solução:



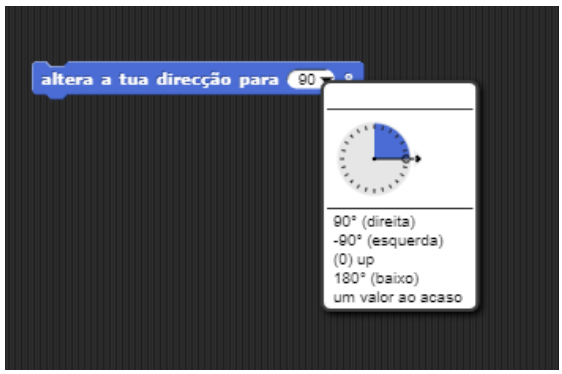
[Passo 3]

Abre um novo projeto e importa os trajes da Avery. Adiciona um cenário apropriado para a Avery andar. Cria uma animação, colocando-a a andar do lado esquerdo para o direito do *stage*. Tenta descobrir como a animar de forma a que os seus passos pareçam ligados uns aos outros, como na vida real.

Possível solução:

[Passo 4]

Até agora, escreveste programas em que o *sprite* apenas se movia numa direção. Nesta tarefa, terás de rodar o rato, de forma a alcançar o queijo. Para o fazer, podes escolher:



a) Dizer-lhe a direção na qual tem de olhar

b) Podes dizer-lhe para se virar num determinado ângulo, na direção dos ponteiros do relógio 

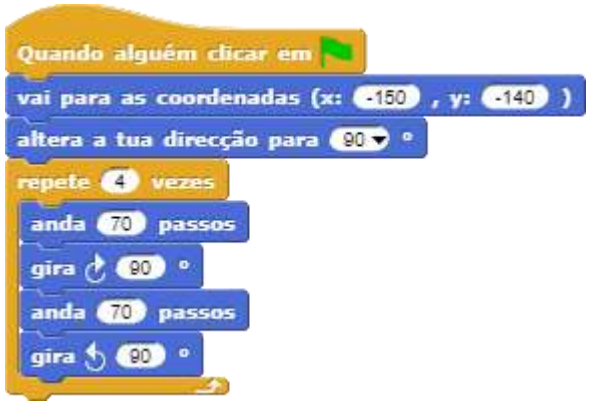
ou na direção contrária 

. Um círculo completo tem 360 graus, por isso, se quiseres virar-te para a direção oposta à qual estás agora, viras 180 graus. Se quiseres virar para a tua esquerda, escolhes rodar 90 graus na direção contrária à dos ponteiros do relógio. Para virar para a direita, rodas 90 graus na direção dos ponteiros do relógio.

Abre https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese. Escreve o programa que o rato terá de seguir para alcançar o queijo andando só na zona verde. Aponta para a direção para a qual se dirige e usa o bloco anda ___ passos. Para ver como o rato se move, usa o "espera 1 segundo" entre as linhas.

Solução:

Agora tenta escrever um programa com rotação de 90 graus.



[Passo 5]

Como viste, o rato moveu-se em várias direções para chegar ao queijo. Por vezes, é normal que não queiras que o teu *sprite* fique virado ao contrário, mas apenas que rode para a esquerda ou direita. Para certificar de que ele se move para onde queres, tens de clicar no ícone apropriado à esquerda do teu *sprite*.

A seta circular significa que o teu *sprite* pode rodar em todas as direções (como o rato). A seta <-> significa que o teu *sprite* se irá mover apenas para a esquerda/direita (é este que deves selecionar para evitar que o cão caminhe sobre a sua própria cabeça).

A última seta, ->, significa que o *sprite* se mantém exatamente como

	está (podes usar este para o macaco). Tenta reescrever os teus programas para o cão e o macaco, para que eles se dirijam primeiro até ao objeto e depois regressem, virando-se. Presta atenção e escolhe o modelo de rotação mais apropriado.
Instrumentos e recursos para o Professor	● Soluções para o programa da bailarina: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dancing ● Avery a andar: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Avery_walking ● Solução para encontrar o queijo: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese_solution
Recursos/Materiais para os alunos	● Encontra o queijo https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese ● Instruções para o estudante (C4G4_InstructionsForStudent.docx)

Cenário de aprendizagem 5 – Sons da quinta

Título do cenário de aprendizagem	Sons da quinta
Experiência prévia de programação	● O estudante é capaz de adicionar um cenário ● O estudante é capaz de criar um *sprite*. ● O estudante sabe colocar o *sprite* a falar.
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Adicionar som a partir da biblioteca de media do Snap! ● Importar som a partir de outras plataformas de média. ● Gravar um novo som ● Reproduzir som quando uma tecla é pressionada Objetivos específicos de aprendizagem, orientados para o pensamento algorítmico: ● O estudante adicionar som a partir da biblioteca do Snap e consegue reproduzi-lo quando uma tecla é pressionada ● O estudante importa som a partir do computador e consegue reproduzi-lo pressionando uma tecla. ● O estudante grava um novo som e consegue reproduzi-lo pressionando uma tecla.
Objetivo, tarefa e breve descrição das atividades	**Breve descrição:** Jogo simples de programação em que o jogador aprende os sons de animais pressionando certas teclas. **Tarefas**: Como primeiro passo, o estudante tem de escolher um cenário para a cena. Depois, o estudante tem de programar a camponesa para dizer as instruções: 1) se queres ouvir o cão, clica na tecla “D”!; 2) Se queres ouvir a vaca, clica na tecla “C”; 3) se queres ouvir a ovelha, clica na tecla “S”; 4) se queres ouvir o porco, clica na tecla “P”; 5) se queres ouvir o cavalo, clica na tecla “H”! Depois disso, o estudante tem de programar a tarefa ditada pela camponesa. **Objetivo**: Será ensinado aos estudantes como adicionar um novo som e como o utilizar. Irão também aprender a usar o bloco do som (“toca o som [*nome do som*]”) e o bloco de controlo (“quando a tecla [*tecla*] é pressionada”).

Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Active learning, game-design based learning
Formas de ensino	Ensino presencial Trabalho individual
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) Motivação-Introdução Motivamos os alunos através do jogo (eles não vêm o código). O objetivo da  lição é fazer com que o jogo fique assim: [Passo 1] O primeiro passo é escolher o cenário do jogo. Este deve ter vários animais. Temos três opções: 1. Os estudantes desenham os seus próprios cenários; 2. Os estudantes procuram imagens gratuitas online; 3. Nós disponibilizamos os cenários para os estudantes (se quisermos poupar tempo).

Os estudantes já sabem como adicionar um cenário, por isso, podem fazê-lo

individualmente.

[Passo 2]

O segundo passo é adicionar a camponesa. Tal como no passo anterior, temos três opções:

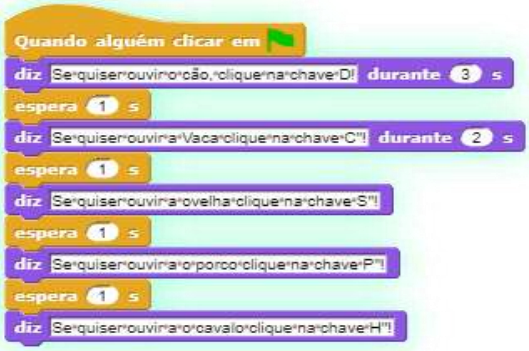
1) Os estudantes desenham eles próprios a camponesa;
2) Os estudantes procuram imagens gratuitas online;
3) Nós disponibilizamos a imagem da camponesa aos estudantes (se quisermos poupar tempo).

Os estudantes já sabem como adicionar um novo *sprite*, por isso podem

fazê-lo individualmente.

[Passo 3]

A seguir, os alunos têm de programar as instruções para o jogador. As instruções são dadas pela camponesa. Os alunos podem fazê-lo usando os blocos *Aparência/diz […]* e *espera [n]*. Os alunos também já sabem concretizar este passo, por isso podem fazê-lo sozinhos.



Implementação

A seguir, mostramos aos estudantes como adicionar som ao jogo. Temos três opções:

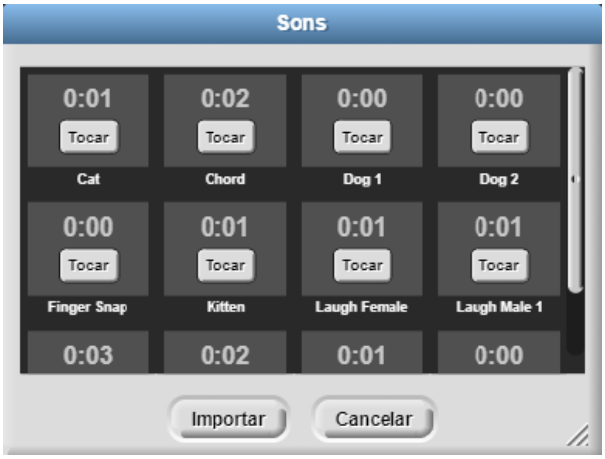
1. Importar um ficheiro de áudio da biblioteca do Snap;
2. Importar um som do próprio computador, arrastando-o até ao Snap!;
3. Gravar um novo som no Snap!

As três opções são mostradas aos estudantes através de ensino presencial. Depois de serem apresentadas, os estudantes podem começar a programar as tarefas seguintes, individualmente (com o apoio do professor).

[Passo 4]

Os estudantes têm de programar o som do cão. Quando o jogador pressionar a tecla “D”, o cão tem de ladrar. O estudante deve, em primeiro lugar, importar o som a partir da biblioteca do Snap! para o separador de som de fundo.

De seguida, escolhem o som que pretendem (Cão 1 ou Cão 2).

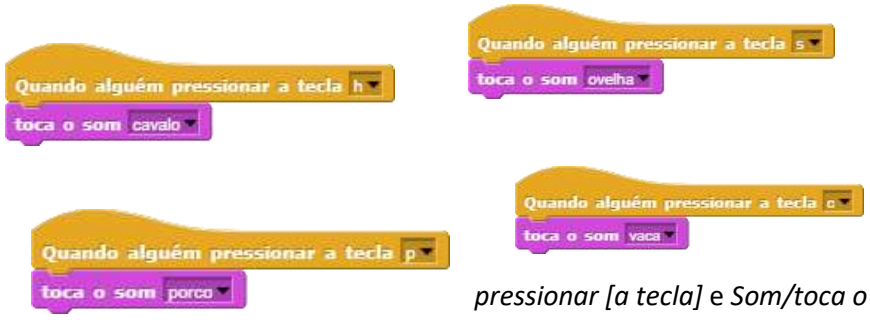


Os estudantes têm de programar o som do cão para ser reproduzido quando a tecla "D" for pressionada. Podem fazê-lo usando os blocos *Controlo/quando alguém pressionar [a tecla]* e *Som/toca o som [nome do som].*

[Passo 5]

Os estudantes têm de programar os sons dos animais. Primeiro, têm de adicionar sons a partir dos seus computadores. Podem fazê-lo arrastando os sons para o separador do som.

Depois de os sons terem sido importados, podemos clicar nestes com o lado direito do rato para mudar os seus nomes. No nosso caso, eles são “vaca”, “porco”, cavalo” e “ovelha”. Depois, os estudantes têm de adicionar os sons aos guiões. Podem fazê-lo usando o bloco *Controlo/quando alguém*



pressionar [a tecla] e *Som/toca o som [nome do som]*.

[Passo 6]

O próximo passo é programar a mensagem de boas vindas da camponesa. Quando o utilizador começar o jogo, a camponesa tem de dizer “Bem-vindo/a à minha quinta”. Em primeiro lugar, os estudantes têm de gravar esta mensagem. Para fazê-lo, usam o gravador de som (botão vermelho) localizado no separador de Som (da camponesa). Depois de gravar o som, têm de o guardar (botão Guardar).

Depois de o som estar guardado, também podemos alterar o seu nome (clicando com o lado direito do rato). No nosso caso, chama-se “quinta”.

Agora, os estudantes têm de adicionar som aos guiões da camponesa. Para tal, usam o bloco *Som/toca o som [nome do som]*.

[Tarefa adicional]

O aluno pode atualizar a quinta da forma que quiser, adicionando novos *sprites* (camponês, galinhas, tratores…) e sons.

Reflexão e avaliação

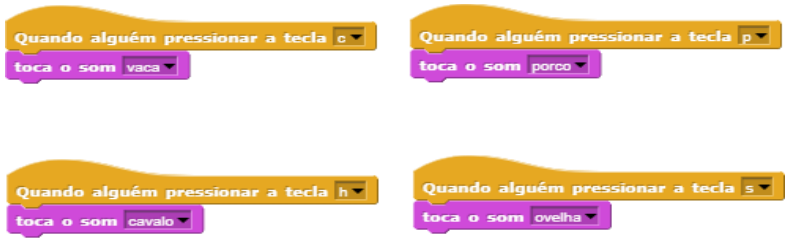
Os estudantes devem resumir:

a) Como adicionaram sons ao código;

b) Quais foram os blocos utilizados para introduzir som no código;

c) Quais foram os blocos de controlo que utilizaram no código;

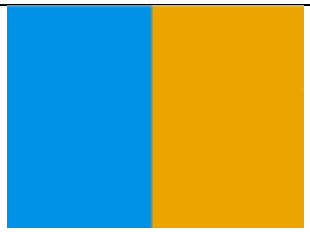
d) Como e quando utilizaram blocos de som e blocos de controlo.

	[Código final] *A camponesa*  *O cenário* 
Instrumentos e recursos para o Professor	• Whole activity in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Farm • Website of free images: https://pixabay.com/ • Website of free sounds: https://www.zapsplat.com/ • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para os alunos	• Template in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Sounds%20of%20the%20farm_0 • Website of free images: https://pixabay.com/ • Website of free sounds: https://www.zapsplat.com/ • Instructions for student (C4G5_InstructionsForStudent.docx)

Cenário de aprendizagem 6 – As férias de verão do camaleão

Título do cenário de aprendizagem	As férias de verão do camaleão
Experiência prévia de programação	Não é requerida experiência prévia de programação
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● movimento do objeto baseado em eventos; ● sensores de cores singulares ou múltiplos; ● leitura de valores Booleanos em expressões lógicas; ● definir, diferenciar, verificar dinamicamente e responder a diferentes estados de jogo; Objetivos específicos de aprendizagem, orientados para o pensamento algorítmico: ● o estudante movimenta objetos com setas do teclado usando eventos, e tem em conta as restrições; ● o estudante usa um bloco de sensor de cor para obter o valor booleano para a leitura de sensores de cores singulares ou múltiplos; ● o estudante percebe que o estado do objeto pode ser expresso com as cores que o objeto toca, ● o estudante sabe distinguir entre dois estados (básicos) e cinco estados (completos) e sabe como expressá-los através de expressões lógicas, ● o estudante percebe que a posição do objeto muda dinamicamente e usa o *loop* infinito para repetidamente verificar o estado atual, ● o estudante usa a condicional if para dar respostas diferentes com base na posição atual do objeto
Objetivo, tarefa e breve descrição das atividades	**Breve descrição:** Programar um jogo simples em que o objeto irá mudar de traje com base na cor do cenário **Tarefas:** Os estudantes têm de programar o camaleão para mudar o seu aspeto (traje) e dizer onde se encontra em cinco situações diferentes: 1) quando estiver a nadar no mar, tem de mudar a sua cor para azul e dizer “Estou a nadar no mar”; 2) quando estiver entre o mar e a praia, a sua pele muda para metade azul, metade cor de areia, e ele deve dizer “estou entre a praia e o mar”, 3) na praia, ele

	fica cor de areia e diz “estou a relaxar na praia”, 4) entre a praia e a floresta, ele fica metade verde, metade cor de areia e diz “estou entre a praia e a floresta”, 5) na floresta, a sua pele fica verde e ele diz “estou a refrescar-me à sombra da árvore”. **Os estudantes irão conhecer o bloco de sensores de cores e aprender como o utilizar em expressões lógicas para distinguir entre estados dinamicamente em mudança e dar as respostas adequadas.**
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem colaborativa, resolução de problemas
Formas de ensino	Ensino presencial Trabalho individual/trabalho em pares/trabalho de grupo
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) O camaleão foi de férias de verão. Ele gosta de dar mergulhos no mar, relaxar na praia e, quando está demasiado calor, gosta de se abrigar à sombra de uma árvore. Como é um camaleão, muda de cor de acordo com o cenário em que se encontra. [Versão básica] Na versão básica, temos de distinguir entre dois estados: [Passo 1] Pedimos aos estudantes para editar o cenário de forma a que fique dividido em duas partes, uma cor de areia e a outra azul, cada uma delas representando um local diferente. O azul corresponde ao mar, e a cor de areia corresponde à praia. Podemos dizer aos estudantes para incluírem outros elementos para tornarem o cenário mais realista, como ondas, castelos de areias, chapéus de sol, etc… Devem ter em atenção que não devem escolher itens muito grandes e completamente de cores diferentes do que o cenário, porque nesse caso o bloco de sensor de cor não será capaz de reconhecer em que parte do cenário a personagem se encontra.



[Passo 2]

Têm de desenhar um camaleão e pintar a sua pele de duas cores diferentes

[Passo 3]

Primeiro, têm de fazer, usando as teclas, com que o seu camaleão se mova em quatro direções diferentes. Podem escolher a sua própria combinação (por exemplo, teclas de setas ou WASD). A este ponto, é expectável que, com base em atividades anteriores, os estudantes já o saibam fazer. É importante relembrar o estudante que a personagem irá sair do *stage* se não utilizarem o bloco correto ao programarem o movimento (saltar se estiver na borda do bloco).

Para fazer com que o camaleão se mova de forma um pouco mais realista, queremos que ele se vire para a esquerda e direita na direção horizontal para a qual estamos virados (usando o bloco *aponta em direção a*).

[Passo 4]

Apresentamos aos estudantes a possibilidade de a personagem ter sensores para as cores em que toca. Com o bloco “estás a tocar na cor?” podemos obter a informação, na forma de valores booleanos – verdadeiro ou falso se ele estiver a tocar em determinada cor. Como recebemos valores booleanos neste bloco, podemos usá-lo para a condicional if, onde é decidido se queremos executar comandos listados no seu corpo ou não.

De seguida, discutimos com os estudantes quais serão as diferentes posições do camaleão na cena, e como podemos expressá-las usando o bloco de bloco *“estás a tocar na cor?”*

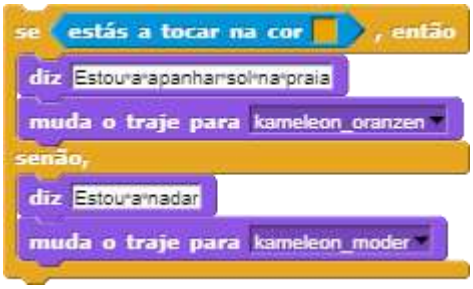
Existem duas formas:

1. Ele está a tocar no azul -> estás a tocar na cor [azul]?
2. Ele está a tocar na cor de areia -> estás a tocar na cor [cor de areia]?

Quando o camaleão toca numa determinada cor, temos de mudar a sua aparência e também temos de o fazer dizer o local onde se encontra. Podemos alterar a aparência do *sprite* alternando entre os seus vários trajes. Isto é feito através do bloco *Aparência/mudar o traje para [opção]* onde podemos selecionar o aspeto que pretendemos. Para fazer com que o camaleão fale, usamos o bloco *Aparência/diz [texto].*

Como existem duas possibilidades, podemos usar o bloco condicional "se –, então"

Podemos escolher a cor que queremos verificar, e automaticamente a outra cor ficará no "então". Como amostra, escolhemos a cor de areia:



[Passo 5]

Para situações em que temos de executar certos comandos durante todo o programa, usamos o *loop* infinito. Tudo aquilo que fica sob o "*loop* infinito" irá ser executado repetidamente. Neste caso, podemos dizer aos alunos que é exatamente isto que pretendemos para criar este jogo.

[Código final]

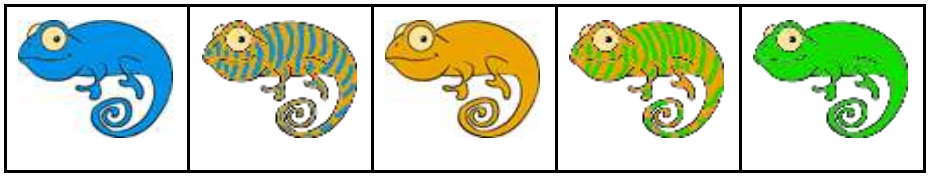
[Versão completa]

[Passo 1]

Pedimos aos estudantes para editar o cenário de forma a que fique dividido em três partes de cores diferentes, cada uma delas representando um local diferente: azul é para o mar, cor de areia para a praia, verde para a floresta. Eles podem também adicionar outros itens para tornar os locais mais realistas: ondas, conchas, castelos de areia, chapéus de sol, árvores, etc.…, desde que tenham em atenção que não devem escolher itens muito grandes e completamente de cores diferentes do que o cenário, porque nesse caso o bloco de sensor de cor não será capaz de reconhecer em que parte do cenário a personagem se encontra.

[Passo 2]

Os estudantes têm de desenhar um camaleão e pintar a sua pele com cinco combinações diferentes, representando a sua posição na cena:



[Passo 3]

Primeiro, têm de fazer, usando as teclas, com que o seu camaleão se mova em quatro direções diferentes. Podem escolher a sua própria combinação (por exemplo, teclas de setas ou WASD). A este ponto, é expectável que, com base em atividades anteriores, os estudantes já o saibam fazer. É importante relembrar aos alunos que se a personagem irá sair do *stage* se não utilizarem o bloco correto ao programarem o movimento (saltar se estiver na borda do bloco).

Para fazer com que o camaleão se mova de forma um pouco mais realista, queremos que ele se vire para a esquerda e direita na direção horizontal para a qual estamos virados (usando o bloco *aponta em direção a*).

[Passo 4]

Apresentamos aos estudantes a possibilidade de a personagem ter sensores para as cores em que toca. Com o bloco “estás a tocar na cor?” podemos obter a informação, na forma de valores booleanos – verdadeiro ou falso se ele estiver a tocar em determinada cor. Como recebemos valores booleanos neste bloco, podemos usá-lo para a condicional if, onde é decidido se queremos executar comandos listados no seu corpo ou não.

De seguida, discutimos com os estudantes quais serão as diferentes posições do camaleão na cena, e como podemos expressá-las usando o bloco “estás a tocar na cor?”

Há 5 posições possíveis:

1. Ele está completamente na parte azul -> estás a tocar na cor? [azul]
2. Ele está entre a parte azul e a parte da areia -> estás a tocar na cor [azul] E estás a tocar na cor [cor de areia]
3. Ele está completamente na parte da areia -> estás a tocar na cor? [cor de areia]
4. Ele está entre a parte da areia e a parte verde -> estás a tocar na cor [cor de areia] E estás a tocar na cor [verde]
5. Ele está completamente na parte verde -> estás a tocar na cor? [verde]

Quando o camaleão encosta numa determinada cor, temos de mudar a sua aparência e também temos de o fazer dizer o local onde se encontra. Podemos alterar a aparência do *sprite* alternando entre os seus vários trajes. Isto é feito através do bloco Aparência/mudar o traje para [opção] onde podemos selecionar o aspeto que pretendemos. Para fazer com que o camaleão fale, usamos o bloco *Aparência/diz [texto].*

Em primeiro lugar, tratamos das situações mais simples, em que o camaleão é totalmente da mesma cor do que a cena:

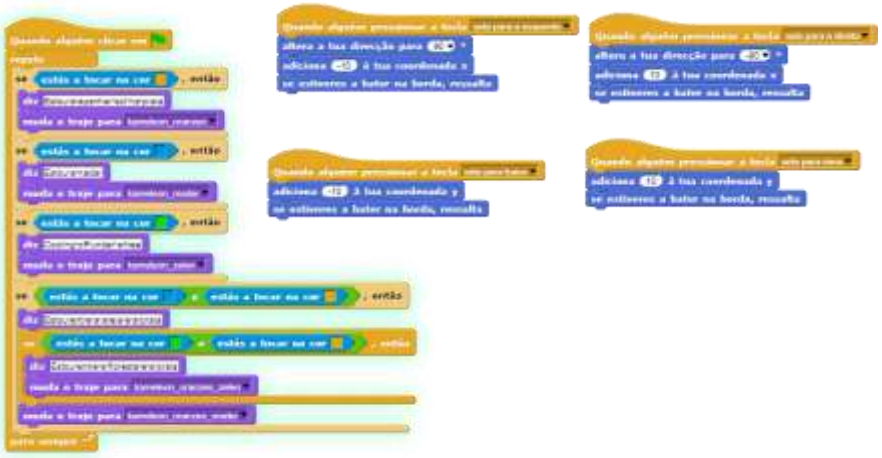
De seguida, formamos uma expressão lógica, com o uso do operador lógico E, porque queremos verificar se o camaleão está a tocar duas cores ao mesmo tempo:


Se combinarmos as frases condicionais acima e as colocarmos sobre o bloco *Quando a bandeira verde é clicada*, notamos que essas condições estarão selecionadas apenas uma vez. Temos de chamar à atenção para tal porquê controlamos o movimento da personagem principal, e a posição do camaleão estará a mover-se durante o jogo todo. Por isso é que temos de verificar se estas condições estão selecionadas, não apenas uma vez, mas durante o jogo completo!

[Passo 5]

Em situações nas quais temos de executar comandos durante toda a duração do programa, usamos o *loop* infinito. Tudo o que estiver escrito sob o *loop* infinito vai ser executado repetidamente. Neste caso, podemos dizer aos alunos que é exatamente isto que pretendemos para criar este jogo.

[Código final]



[Estudantes ajustam o código]

Para simplificar esta atividade, podemos preparar parte do código previamente num template e pedir aos estudantes que o completem.

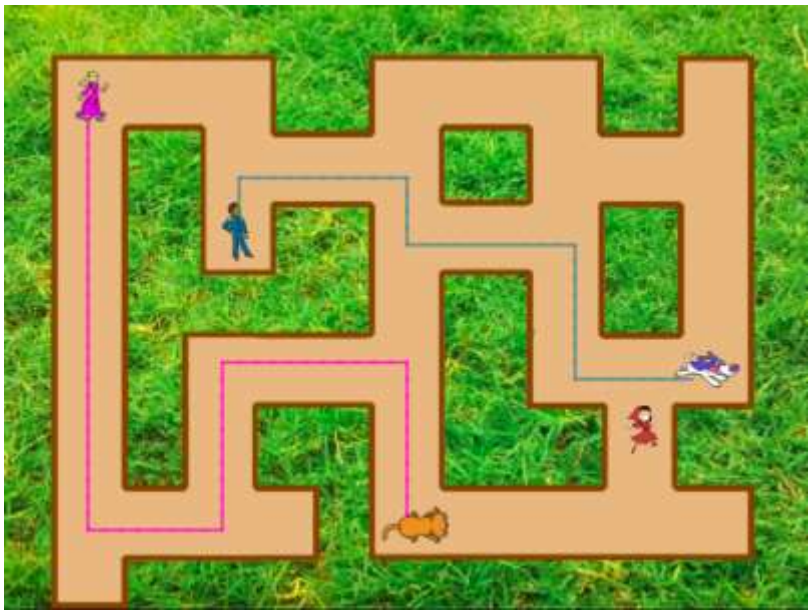
Os estudantes que seguiram o caminho de aprendizagem sugerido já aprenderam a mover objetos usando teclas. Podem, por isso, incluir o código de movimento no template. Podem modificar as definições das teclas, de teclas de setas para personalizadas (WASD, por exemplo).

	Para ajudá-los a compreender o conceito de *loop* infinito e perceber como o utilizar para verificar a cor do cenário, podes incluir código para detetar duas situações: 1) o objeto é inteiramente de uma só cor, 2) o objeto toca duas cores ao mesmo tempo. Pedimos aos estudantes que completem o código para ambos os casos. Template sugerido: 
Instrumentos e recursos para o Professor	• Atividade completa no Snap!: Básico: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_simple Completo: https://snap.berkeley.edu/project?user=zapusek&project=chameleon • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para os alunos	• Template no Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_template • Atividade pré-construída no Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_half_baked

	• Instruções para os estudantes (C4G6_InstructionsForStudent.docx)

Cenário de aprendizagem 7 - Ajudar o Príncipe e a Princesa a encontrarem os seus animais

Título do cenário de aprendizagem	Helping Prince and Princess to find their animals
Experiência prévia de programação	Adicionar texto ao *sprite* Movimento de objetos com as setas usando eventos Usar o condicional para *estás a tocar* para o estado do objeto Usar eventos
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Condicionais para *estás a tocar [cor]* ● Movimento seguindo coordenadas ● Levanta a tua caneta, baixa a tua caneta ● Cor da caneta Objetivos específicos de aprendizagem orientados para o pensamento algorítmico: ● O estudante usa a condicional "if" para o estado dos objetos e coloca o objeto de volta, se tocar determinada cor ● O estudante determina coordenadas x e y iniciais para o *sprite* ● O estudante levanta e baixa a caneta para desenhar uma linha/caminho ● O estudante muda a cor da caneta dependendo do par que está a conectar ● O estudante apercebe-se que no início tem de apagar todos os caminhos anteriores
Objetivo, tarefa e breve descrição das atividades	**Breve descrição:** Os estudantes têm de ajudar a Princesa a encontrar o seu gato e o Príncipe a encontrar o seu cão. Para tal, têm de ir até à Princesa e mostrar-lhe, desenhando uma linha, o caminho até ao seu gato, fazendo depois o mesmo com o Príncipe. Os estudantes têm de evitar encontros entre os dois animais, para que os caminhos não se cruzem. **Tarefas:** Em primeiro lugar, os estudantes têm de escolher o cenário apropriado (labirinto). Depois adicionam cinco *sprite*s ao labirinto (o seu *sprite* (uma rapariga), uma princesa, um príncipe, um cão e um gato. Depois, têm de programar movimento com setas (usando eventos) para a rapariga, tendo em atenção que a imagem não pode pisar a relva. Depois, devem programar para desenhar com uma caneta e mudar a cor da caneta usando eventos. Além disso, também devem programar o evento inicial, que limpa o caminho, e a rapariga que dará as instruções.

	Objetivo: Os estudantes irão aprender a desenhar através do movimento com as setas. Além disso, irão aprender a usar condicionais para impedir que o *sprite* se mova pelo ecrã todo.
Duracao da Atividade	30 minutos
Estratégias e métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design, resolução de problemas
Formas de ensino	Aprendizagem Presencial Trabalho individual
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) Logo no início, é dado aos estudantes: ● Cenário ● *Sprite* da rapariga ● Código do movimento numa direção A rapariga decide ajudar a Princesa a encontrar o seu gato e o Príncipe a encontrar o seu cão. Para tal, irá desenhar os caminhos a percorrer até aos animais. Para evitar confusão, os caminhos devem ser de cores diferentes e não devem cruzar-se.  [Passo 1] Nós pedimos aos alunos que editassem o cenário de plano de fundo - um

labirinto. Para a implementação do “se estás a tocar na cor” ou o cenário (relva) deve ser monocromático ou o caminho deve-se ter uma moldura monocromática, como no nosso caso. Para evitar esses problemas com encontrar o cenário apropriado nós disponibilizamos o cenário.

[Passo 2]

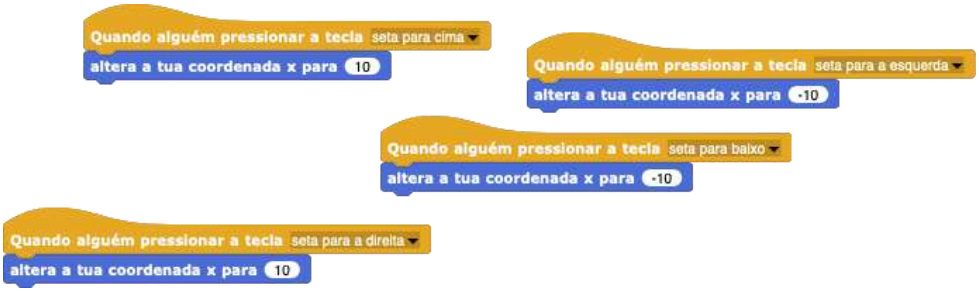
Os estudantes já têm a rapariga no início. Eles precisam encontrar outros quatro sprites (imagens) e colocá-los no labirinto. Para todos os sprites (imagens) deve-se definir o tamanho adequado (que deve ser menos que a largura dos caminhos do labirinto).

Para cada imagem usa-se o código:

O tamanho recomendado para a rapariga é 8%, os outros sprites (imagens) podem ser maiores.

[Passo 3]

Depois eles tem que fazer o movimento da rapariga em quatro direções utilizando as teclas. Assumimos que os estudantes já saibam fazer isso devido as aulas anteriores. De qualquer modo nós damos o código de uma direção, que os ajudam a construir os outros três.



[Passo 4]

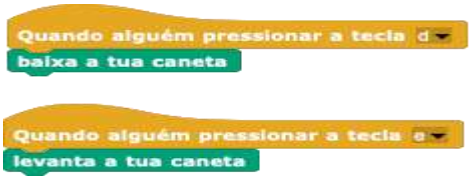
No próximo passo tem que se prevenir o movimento das raparigas pelo campo. Se faz isso adicionando uma barreira condicional ao tocar na cor castanha. Se a rapariga estiver a tocar na cor castanha (fim do caminho), ela volta 10 passos para trás. Nós não vemos aqueles dois passos e é como se a rapariga continuasse na mesma posição. Esse é um código para mover-se para a direita, portanto 10 passos para trás significa trocar x por -10.

Adiciona-se esse código em baixo do código anterior, por exemplo, para a seta da direita:

É necessário fazer o mesmo para as outras 3 direções.

[Passo 5]

Em seguida, deve-se programar o desenho. Se faz isso pelos blocos "levanta a tua caneta" e "baixa a tua caneta" utilizando " *Quando alguém pressionar a tecla...*".

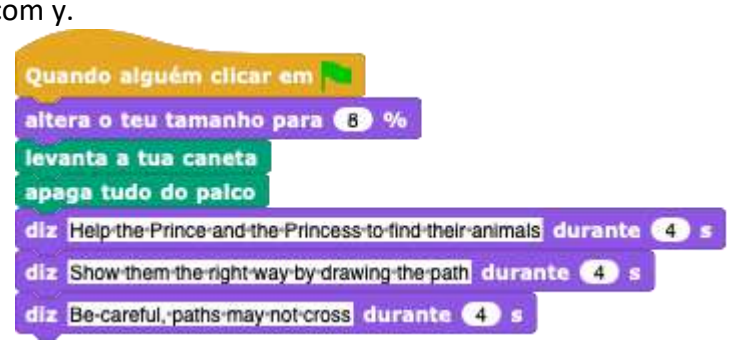


Quando a tecla "D" é pressionada e a rapariga se move, ela desenha uma linha. Quando a tecla "E" é pressionada, o desenho para.

Da mesma maneira escolhe-se a cor da caneta ao pressionar a tecla.

[Passo 6]

Por fim, os estudantes programam o comando “quando se clica na bandeira verde”, onde adicionam algumas instruções que a rapariga diz no início.
Ao jogar, para-se e joga-se novamente, os estudantes verão que é bom adicionar os seguintes blocos: levanta a tua caneta (caso estiver permanecido abaixo do jogo anterior), *limpar* (limpa-se o caminhos feitos nos jogos anteriores) e ir para x, y (a rapariga sempre começa nas determinadas coordenadas, que estão dentro do caminho e não na relva). Para determinar as coordenadas de início para as raparigas, apanha-se a rapariga com o rato e coloca-se no local que deseja-se que ela inicie. Depois clica-se nos blocos de movimentação, onde encontra-se a posição x e y. Ao clicar na posição x descobre-se a posição x da rapariga, ocorre o mesmo com y.



[Código Final]

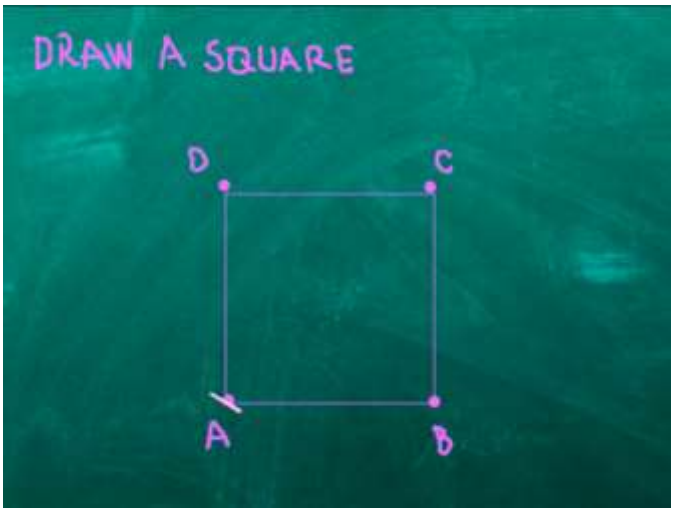
Rapariga

	Exemplo Princesa:  [Tarefas adicionais] Os estudantes podem adicionar as tarefas extras se desejarem ou podem seguir as tarefas abaixo: ● Determine as coordenadas iniciais para o Príncipe e para a Princesa e escreva o código do movimento deles. Determine o tamanho apropriado para eles. Eles devem desenhar um caminho para os animais deles. ● Adicione outro sprite (animal) para a rapariga. ● Cada sprite deve desenhar com uma cor diferente. ● Ajuste as instruções iniciais. Adicione instruções para mover um sprite e desenhar ao clicar no sprite. Por exemplo, a Princesa diz: "Move-me pressionando as teclas W, S, A e D. Eu desenho o caminho ao pressionar a tecla 3. Eu paro de desenhar ao pressionar a tecla 4. Ajuda-me encontrar meu gato!".
Instrumentos e recursos para o Professor	● Atividade completa em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals ● Atividade em Snap! com tarefas adicionais (possível solução): https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals%20%2B%20Add.%20Task ● Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Objetivo, tarefa e breve descrição das atividades	● Atividade pré-construída no Snap!: https://snap.berkeley.edu/project?user=mateja&project=Helping%20Princ e%20and%20Princess%20to%20find%20their%20animals%20-%20Part ● Instruções para os alunos (C4G7_InstructionsForStudent.docx)

Cenário de Aprendizagem 8 - Desenhar com giz

Título do cenário de	Desenhar com giz

Aprendizagem	
Experiência Prévia de programação	Adicionar texto para o *sprite* Desenhar com caneta (*pen up, pen down*, definir cor) Mover-se com os passos Utilizar *loops* Utilizar eventos
Objetivos de Aprendizagem	Objetivos Gerais de Aprendizagem: ● Repetir o *Loop* ● Virar para 90 graus ● Apontar em direção ● Mudar cenário Objetivos de aprendizagem específicos orientados por pensamento algorítimico: ● Estudantes usam *loop repeat* quando os mesmos blocos repetem-se 2/4 vezes ● Estudantes usam virar para 90 graus quando desenham diferentes formatos (quadrado, retângulo e a letra "T") ● Estudantes percebem o significado de apontar em direção 90 ● Estudantes sabem como mudar o cenário em combinação com um evento quando a tecla é pressionada
Objetivo, Tarefa e uma breve descrição das atividades	Descrição breve: O jogador recebe três planos de fundo diferentes e tem que conectar pontos em três formatos diferentes - um quadrado, um retângulo e a letra "T". Tarefas: Os estudantes escolhem o fundo “boardS” e iniciam a desenhando um quadrado. O ponto de início é o ponto "A". Enquanto desenham o quadrado, certos passos repetem-se 4 vezes, portanto ao invés de escrever o mesmo código 4 vezes, pode-se utilizar um *loop repeat* 4 vezes. Depois desenha-se um retângulo, também utilizando um loop repeat, dessa vezes repete-se 2 vezes. Na última tarefa deve-se conectar os pontos no formato da letra "T", onde deve-se descobrir o número de passos. Pode-se usar o *loop repeat* quando possível. **Objetivo: Estudante serão introduzidos a desenhar diferentes formas com um código. Eles aprenderão a usar o loop repeat para encurtar o código e mudar o cenário.**
Duração das atividades	60 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem Ativa, Aprendizagem através de game-design, resolução de problemas

Formas de Ensino	Trabalho presencial Trabalho Individual/ Trabalho em pares
Sumário da Lição	(Motivação-Introdução, Implementação, Reflexão e Avaliação) Inicialmente é dado aos estudante: ● Três planos de fundo com todas os pontos que devem ser conectados ● Imagem de um giz O giz quer desenhar um quadrado, um retângulo e conectar os pontos para fazer o formato da letra "T", mas não sabe como se mover e como se virar. Escreva um código e mostre ao giz como fazer isso! [Passo 1]  Estudantes iniciam com esse cenário. Eles escrevem um código para desenhar um quadrado. Iniciando no ponto "A"” eles movem x passos até o ponto "C", vira 90 graus para a esquerda, move x passos até o ponto "A" (e vira 90 graus para a esquerda).

Utilizar o virar 90 graus é a forma mais fácil, uma vez em que podemos usá-lo sempre (apenas depende se queremos virar para a esquerda ou para a direita). Usar ponto em direção 0º, 90º, 180º, -90º é outra opção, mas é mais complicada de se utilizar porque temos quatro opções distintas e não podemos utilizar o repetir *loop* nas mesmas.
O bloco Espera 1 segundo é adicionado apenas para se ver o desenho / todos os passos. Sem esse bloco o código aconteceria tudo em um segundo. Os estudantes devem testar sem este bloco para conseguir entender o seu significado.
Pergunta-se ao estudante como eles poderiam encurtar o código, se possível. Há alguma parte que repete? A resposta é sim. Ao invés de escrever o mesmo código 4 vezes, dentro da programação utiliza-se repetir *loop*.

Se quisermos ver de facto o que desenhamos temos que colocar um bloco *pen down* antes do *repeat loop*.

Se quisermos que o giz não fique a girar enquanto vira, clica-se em

"don't rotate" em direção ao bloco.

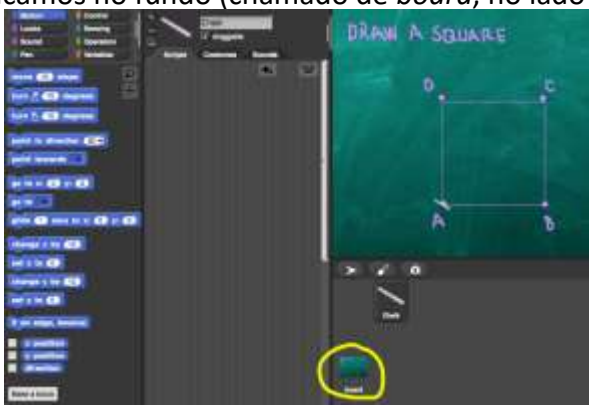
[Passo 2]

Para ativar o código, os estudantes utilizam o bloco de evento, por exemplo, quando a tecla S está pressionada. Pode-se também definir a cor da caneta, e, assim como já sabe-se devido às atividades anteriores, blocos seguintes: levanta a tua caneta (caso tivesse ficado para baixo do jogo anterior), limpar (limpa o desenho do jogo anterior) e vai para as coordenadas x, y (para que o giz permaneça nestas coordenadas).
Por vezes, acontece pararmos o programa durante o jogo e então a imagem vira em uma "direção estranha". Isso é um problema quando o jogo começa novamente, se a imagem virou erradamente, irá, por exemplo, para baixo e não diretamente para o primeiro passo. Para evitar esse problema, adiciona-se um block *point na direção 90º*.

[Passo 3]

Após desenhar um quadrado, queremos desenhar um retângulo. Isso significa que temos mudar o fundo.Fazemos isso em dois passos:

a) Clicamos no fundo (chamado de *board*, no lado direito do ecrã).

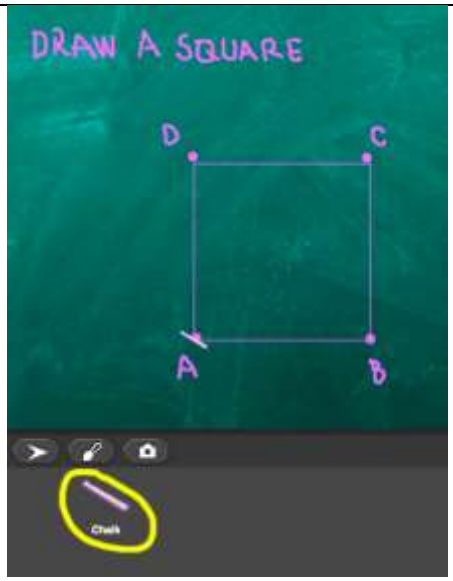


Ao clicar em *planos de fund*o podemos ver todos os três planos de fundo

necessários (*boardSquare, boardRectangle, boardT*), já preparados para essa atividade.

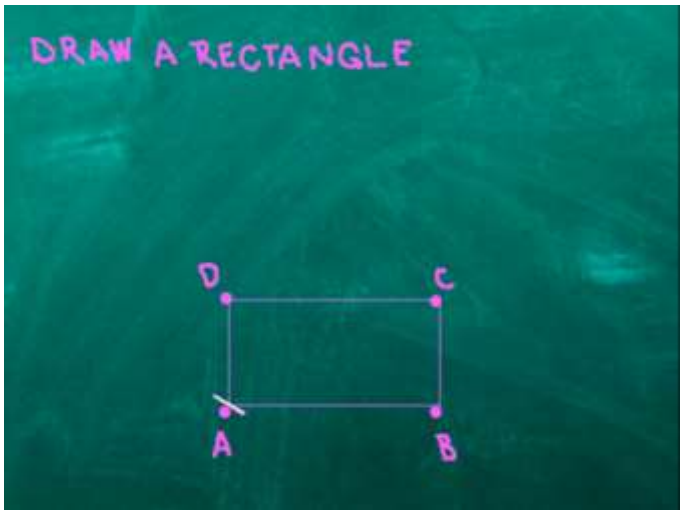
Para programar um código os estudantes devem clicar em *Scripts*. Para programar mudar um cenário escolhido escolhe-se um bloco de eventos quando a tecla R está pressionada e depois troca-se para personalizar *boardRectangle*.

a) Seleciona-se o giz.



Abaixo do código do [Passo 2] os estudantes adicionam um bloco, onde eles dirão ao jogador o que fazer para mudar o cenário, que é, pressionar a tecla "R".

[Passo 4]



Após pressionar a tecla "R", o cenário muda para este. Assim com anteriormente, precisa-se conectar os pontos e desenhar um retângulo. Os estudantes podem copiar o bloco de códigos anterior e corrigi-los para que o programa desenhe um retângulo.

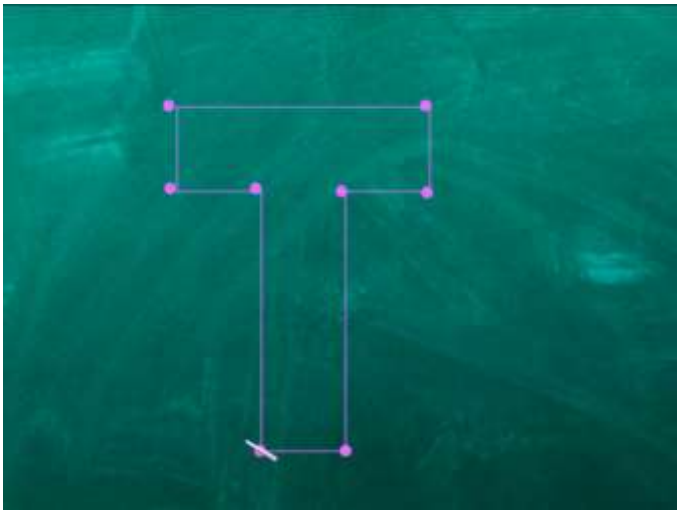
Muda-se a repetição de *loop*. Agora, o loop vai-se repetir 2 vezes.

[Passo 5]

Após desenhar um retângulo, os estudantes irão conectar os pontos no formato de uma letra "T". Isso significa que deve-se mudar o cenário, portanto nesta passo repete-se o [Passo 3], apenas muda-se a letra ("T") e personaliza (boardT):

a) Clica-se no fundo (named board, no lado direito do ecrã), onde devem escrever o código para mudar o cenário. Eles farão isso com *quando a tecla T pressionada* e depois trocar para personalizar *boardT*.

b) Clica-se novamente no giz e abaixo do código do [Passo 4] adiciona-se um bloco, onde dirão ao jogador o que fazer para mudar o cenário, que é, pressionar a tecla "T".

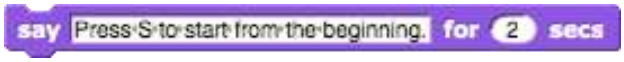
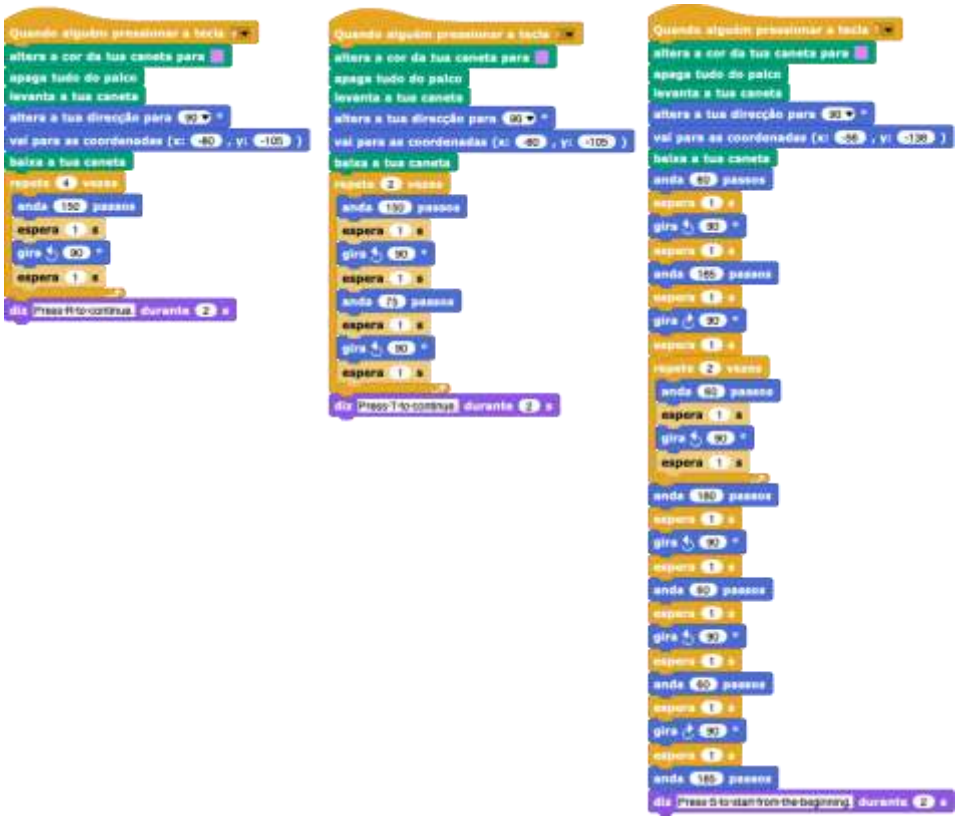
	[Step 6]  Depois de pressionar a tecla "T", o cenárip muda para esse. Assim como no anterior, precisa-se conectar os pontos e desenhar a letra "T"” Os estudantes podem copiar o bloco de códigos anterior e corrigi-los. Os estudantes terão que mudar as coordenadas, que não são as mesmas que as anteriores. Eles já sabem como determinar as coordenadas certas devido a atividade anterior. Depois devem escrever um código para desenhar a letra "T". Eles precisam descobrir o número de passos. Uma solução possível é:

[Passo 7]

Uma vez em que mudamos o cenário, nós podemos retornar ao primeiro cenário para desenhar um quadrado. Então os estudantes terão que adicionar um último código. Deve-se repetir o [Passo 3/5].

a) Clica-se no fundo (named *board*, no lado direito do ecrã), onde eles escrevem um código para mudar o cenário. Eles farão isso com *enquanto a tecla S está pressionada* e depois mudam para personalizar *boardSquare*.

b) Clica-se novamente no giz e abaixo do código do [Passo 6] adiciona-se um bloco, no qual eles dirão ao jogador o que fazer

	para mudar o cenário, que é pressionar a tecla "S".  [Código Final]  [Tarefas adicionais] Os estudantes podem adicionar tarefas extras de acordo com os seus desejos ou podem seguir as tarefas seguintes: ● Adicionar um fundo novo e desenhar alguns pontos. ● Escrever um código que conecta os pontos. Pode-se desenhar um cenário ou utilizar algum já fornecido.
Instrumentos e recursos para o Professor	● Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk ● Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.

Referências/ Materiais para os estudantes	• Atividade pré-construída em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk%20-%20Part • Instruções para os alunos (C4G8_InstructionsForStudent.docx)

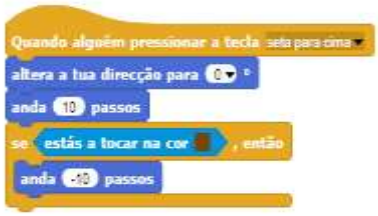
Cenário de Aprendizagem 9 - Apanhar o lixo e limpar o parque

Título do cenário de Aprendizagem	Apanhar o lixo e limpar o parque
Experiência Prévia de programação	Definir as coordenadas iniciais Definir o tamanho do *sprite* Adicionar texto ao sprite Movimento do objeto com a setas de direção Usar as condições *está a tocar em* para alterar o estado do objeto
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● Variáveis ● Mostrar e esconder sprites (imagens) ● Duplicar sprites (imagens) ● Duplicar um bloco de código ● Condições Objetivos de Aprendizagem específicos com base em um pensamento em algoritmo: ● Os alunos usam variáveis para contar o desperdício coletado. ● Os alunos utilizam *esconder imagem* quando a mesma é tocada e *mostrar* a imagem novamente no início do jogo ● Os alunos sabem como duplicar as imagens (por exemplo: de uma garrafa para quatro garrafas). ● Os alunos sabem como duplicar um *bloco de código* (da imagem de uma garrafa a uma imagem de papel). ● Os alunos sabem como utilizar as condições para confirmar se o *sprite* é revelado e se todo o lixo é apanhado.
Objetivo, Tarefa e uma breve descrição das atividades	Breve descrição: O parque está cheio de lixo e a rapariga decide limpá-lo. Após apanhar todo o lixo, ela põe tudo no caixote do lixo. Tarefa: Os alunos devem começar por definir as coordenadas iniciais para a rapariga. O jogo termina quando a rapariga recolher todo o lixo e colocá-lo no caixote de lixo. Para que isso aconteça, os alunos terão de utilizar variáveis para contar os pontos (1 lixo apanhado = 1 ponto). Quando a rapariga toca no lixo, ela recolhe-o, o lixo se esconde e o número de pontos aumenta para 1. Quando ela não cumpre a tarefa de recolher todo o lixo e vai ao caixote de lixo mais

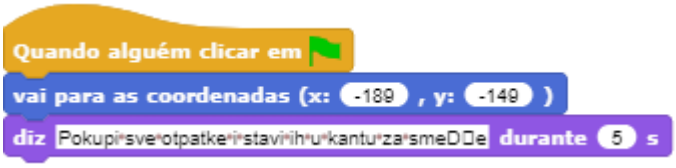
	cedo do que o previsto, o caixote diz que ela deve recolher todo o lixo e só depois voltar. **Objetivo: Os alunos aprenderão como usar variáveis e como duplicar *blocos de código* ou todas as imagens**
Duração das atividades	45 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design e resolução de problemas.
Formas de Ensino	Ensino Presencial Trabalho individual
Sumário da Lição:	(Motivação-Introdução, Implementação, Reflexão e avaliação) É dado aos alunos inicialmente: ● Contexto ● A imagem da Rapariga (com o código de movimento), a imagem da garrafa, a imagem do papel e a do caixote de lixo A rapariga deseja fazer uma caminhada e aproveitar o seu dia no parque. No entanto, quando chega ao parque, encontra-o repleto de lixo, e ela decide recolher todo o lixo. Finalmente, após a limpeza, ela consegue deitar-se na relva limpa e aproveitar o dia de sol. 

[Passo 1]

O fundo é dado, assim como a imagem da rapariga com o código para movimento com chaves e condições caso a linha castanha seja



Os alunos devem definir as coordenadas iniciais para que a rapariga seja capaz de ir do bloco x a y. As coordenadas são escolhidas individualmente, mas é crucial que estejam no percurso. Os alunos já sabem como definir as coordenadas iniciais devido às atividades previamente realizadas. Eles também podem adicionar algumas instruções, por exemplo:



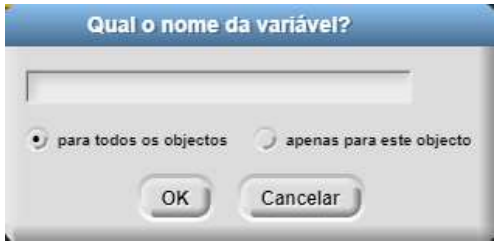
[Passo 2]

Para contar a quantidade de lixo que a rapariga recolheu, iremos usar variáveis.

O que é uma variável ?

A variável é como uma caixa onde guardamos algumas informações. No nosso caso, conseguimos ver a nossa variável como uma caixa , designada por pontuação. Quando a rapariga recolhe um lixo, o lixo é guardado numa variável de pontuação. Esta variável conta quanto lixo a rapariga recolheu.

Como criamos uma variável?



Selecionamos o bloco laranja de "*Variáveis"*, depois selecionamos o botão *Fazer uma "Variável"*, escreva o nome de uma variável e selecione o botão OK. Depois disto, o bloco de pontos aparecerá.

Se a caixa for verificada, a variável com o seu valor estará visível no ecrã.

No início do jogo, o valor da variável tem de ser 0, pois o lixo ainda não foi recolhido. Abaixo do código do [passo 1] o aluno deve adicionar o bloco *"altera*_para_" e atribuir o valor 0. Ao selecionar *drop down* no menu, eles conseguem escolher a variável apropriada, os pontos.

[Passo 3]
Os alunos escrevem o código para a garrafa. A ideia é que a imagem desapareça (o que significa estar escondido) quando toca na rapariga.
O código irá começar quando *a imagem* toca a rapariga. A partir daí temos de pensar em qual situação ela deve apanhar o lixo. Se decidimos que o lixo se esconde quando é recolhido, só é possível apanhar se o lixo ainda estiver lá = estar visível. Se a imagem da garrafa continuar lá, deve ser recolhida e despejada na "caixa da variável". Deste modo, antes havia um total de 0 elementos na variável de pontos, agora há o valor 1. Podemos concluir que ao apanhar o lixo, muda-se o número da variável (pontos) para 1 e aumenta sempre nesse valor. Quando o lixo é recolhido, escondemo-lo.

Agora podemos testar se o código está correto

Selecionamos a bandeira vermelha e recolhemos a garrafa .Deste modo, a garrafa deve desaparecer e o número de pontos deve ser 1.

Depois, ao iniciar uma nova partida do jogo, selecionamos a bandeira vermelha. O que acontece? Onde está a garrafa agora? A garrafa está escondida, escondemos anteriormente. Assim sendo, no início do jogo, devemos programar de modo a que a garrafa esteja visível. Conseguimos fazer isso ao selecionar o bloco *mostra-te*

[Passo 4]

Agora os alunos querem ter mais garrafas no seu jogo para poderem duplicar facilmente a *imagem*. Para isso, devem selecionar com o botão direito do rato na imagem e escolher a opção *duplicar*.

Agora, apenas selecione com o rato a nova garrafa e leve-a até a algum lado dentro do labirinto.

Eles podem repetir este passo e duplicar a garrafa novamente.

[Passo 5]

Agora os alunos querem ter o mesmo código para a *imagem do papel* . Eles conseguem duplicar o código da garrafa ao clicar com o lado direito do rato no bloco de código:

E largá-lo sobre a imagem do papel, ao selecionar com o rato no papel

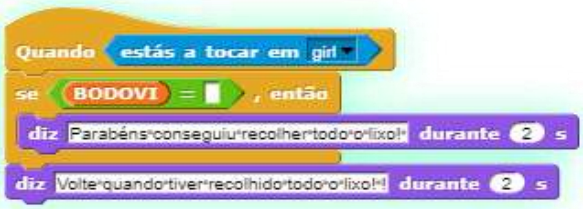
Podem repetir este passo para duplicar o bloco de código quando a Bandeira verde é selecionada – *mostrar*

Também podem repetir [Passo 4] e duplicar toda a imagem para que haja mais lixo de papel no labirinto.

[Passo 6]

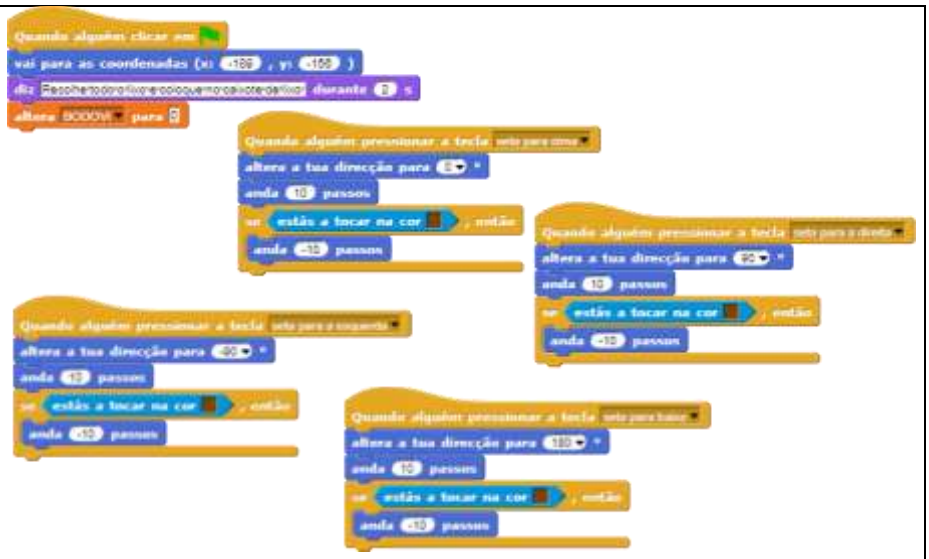
A última coisa que os alunos devem fazer é escrever um código para o caixote de lixo. A imagem do caixote já é dada, eles podem movê-la para qualquer lugar dentro do labirinto.

Para além disso, o código irá ser ativado quando a rapariga o tocar. O caixote terá de verificar se todo o lixo foi realmente recolhido. Graças às *variáveis de pontuação*, isto será fácil de ser conseguido. Digamos que temos 8 tipos de lixo, os alunos terão de verificar se a pontuação equivale a 8. Caso os valores coincidam, significa que todo o lixo foi recolhido, caso contrário não. Eles utilizarão a *condição if* e será adicionado algum texto para informar ao jogador se ele conseguiu recolher todo o lixo ou não.

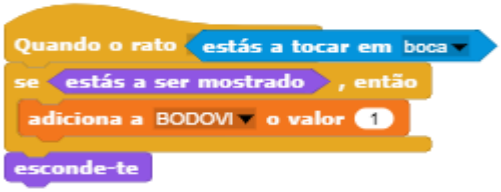


[Código Final]

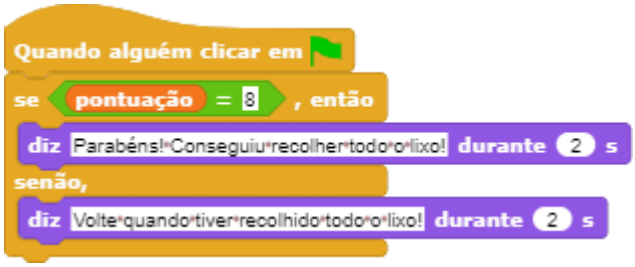
Rapariga



Garrafas / Papéis



Caixote de Lixo



[Tarefas adicionais]
Os alunos podem adicionar tarefas adicionais se quiserem, ou podem seguir as tarefas abaixo indicadas:

- Adicionar outro tipo de lixo. (Por exemplo: Lixo biodegradável).
- O caixote de lixo pode dizer por exemplo: "Recolheste X garrafas, Y papéis e Z melancias".
- Se o jogador ou jogadora apanhar todo o lixo, o caixote poderá dizer: "Parabéns! Conseguiste recolher todo o lixo!"
- Se o jogador ou jogadora não recolher todo o lixo, o caixote poderá informar quais tipos de lixo ainda não foram recolhidos, por exemplo: " Ainda não recolheste todas as

	garrafas", "Ainda não recolheste todas as melancias" e "Volta quando recolheres todo o lixo".
Instrumentos e recursos para o Professor	● Todas as atividades em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park ● Atividades com tarefas adicionais (soluções possíveis): https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20%2B%20Add.%20Task ● Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para alunos	● Atividades Pré-construídas em Snap": https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20-%20Part ● Instruções para o aluno (C4G9_InstructionsForStudent.docx)

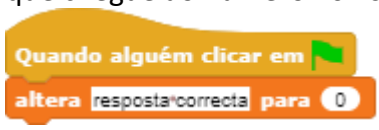
Cenário de Aprendizagem 10 - Alimentar os gatos

Título do cenário de Aprendizagem	Alimentar os Gatos
Experiência Prévia de programação	● Condições (bloco se , então, senão,) ● Imprimir o texto (bloco diz)
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● Definir e aumentar o valor da variável, ● Atribuir uma um valor da variável dentro/fora do loop ● *Loop* (repetir n vezes), ● números aleatórios, ● concatenar uma string, ● operadores: lógica, aritmética, ● Input Objetivos de aprendizagem específicos orientados para um pensamento com base algorítmica. ● Os alunos reconhecem a situação *for* que usa repetir n vezes o loop / ciclo, ● O aluno é capaz de diferenciar entre a atribuição de um valor em todas as interações do *loop/ciclo* e apenas uma vez antes do mesmo. ● Os alunos usam o bloco *input* para obter o número do jogador. ● Os alunos sabem como usar operações aritméticas para gerar a resposta certa. ● Os alunos usam a frase *se - então* para verificar o nível de assertividade da resposta do jogador, e assim, dar a resposta mais apropriada, ● O aluno sabe como usar uma variável para contar as respostas correctas.
Objetivo, Tarefa e uma breve descrição das atividades	Breve descrição: Programar um jogo onde o jogador terá de realizar dez cálculos de multiplicação e contar as respostas correctas. Tarefa: Programar a atividade e a forma como a Marta irá perguntar repetidamente aos jogadores pelo número de gatos que ela consegue alimentar numa determinada sala do abrigo. O número depende do número e tamanho das tigelas. Para cada sala, estes dois números devem ser definidos aleatoriamente. Também é importante temos de ter um contador para contar as respostas correctas. No primeiro abrigo, é importante aparecer a explicação da tarefa para o jogador e apenas depois disso o jogo pode começar. O jogo acaba após a Marta ter perguntado o número de gatos dez vezes. Todas as vezes, ela tem de responder se a resposta dada está correta ou não. No final da atividade, ela tem de explicar resumidamente o nível de sucesso do jogador(a), dizendo quantas vezes o jogador acertou e quantas vezes ele estava errou.

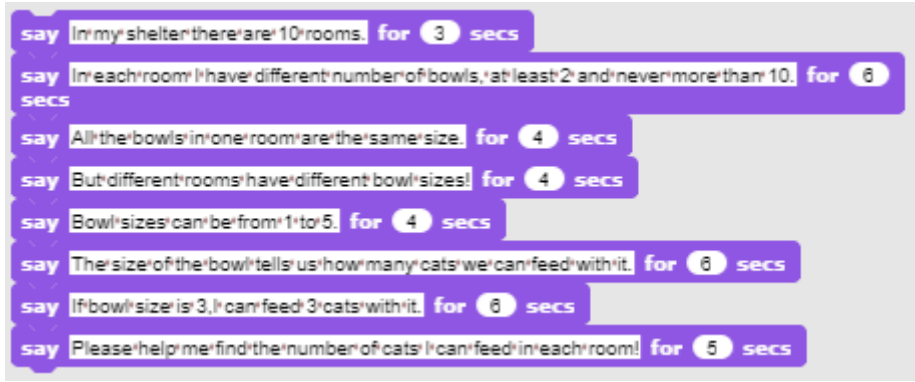
	Os alunos serão introduzidos ao conceito de atribuição múltipla de valores aleatórios variáveis dentro de um *loop*/ciclo e como é visível a diferença de quando fazemos fora do loop/ciclo. Aprenderão também como obter, testar e contar a resposta dos jogadores.
Duração das atividades:	45 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem ativa, colaborativa e com base em resolução de problemas.
Formas de Ensino	Ensino presencial Trabalho individual /trabalho em pares /trabalho em equipa
Sumário da Lição	(Motivação-Introdução, implementação, reflexão e avaliação) O abrigo visa alimentar os gatos em dez salas diferentes da casa. Em cada divisória há um número aleatórios de tigelas (De 2 a 10), de diversos tamanhos (1 a 5) mas dentro de cada sala os tamanhos das tigelas não variam. O tamanho de das tigelas define quantos gatos conseguem comer ali, por exemplo: Se o tamanho da tigela for o 3, significa que 3 gatos conseguem comer ali. Ajude a definir o número de gatos que conseguem ser alimentados em cada divisão do abrigo. [Passo 1] Em primeiro lugar, damos as instruções aos alunos para desenvolverem o design e um background interessante para o jogo. Se quisermos poupar tempo, podemos fornecer previamente o fundo já definido.  [Passo 2]

Temos de selecionar um novo traje para a imagem predefinida da tartaruga que irá representar a guarda do abrigo de gatos.

[Passo 3]
Para guardar os valores necessários precisamos de três variáveis: 1) para guardar o número de respostas correctas, 2) para atribuir um número aleatório de tigelas dentro de cada sala do abrigo (2-10) e 3) para atribuir um número aleatório para a capacidade das tigelas (1-5). O contador das respostas corretas deverá ser definido inicialmente com o valor 0 e os outros dois não precisam ser definidos antes do loop/ciclo, porque iremos atribuir sempre um número aleatório a cada iteração do loop. Também queremos contar os quartos, mas não é necessário uma variável especial para o fazer. Iremos utilizar a mesma variável para o loop. O seu número será definido inicialmente com o valor 1 e aumentará sempre nesse valor para cada iteração até que chegue ao número 10. Isto é replicado da contagem dos quartos.



[Passo 4]
De seguida, temos de programar as instruções para o jogador. Faremos isso através da opção *Aparência/dizer* e *esperar um bloco de n segundos*.



[Passo 5]

Nós discutimos com os alunos quais são as ações que irão acontecer em cada sala e, por tanto, serão as mesmas. Estes são comandos que deverão ser substituídos dentro do bloco do loop para serem executados durante cada iteração do loop/ciclo.

Primeiramente, teremos de atribuir aleatoriamente um valor (1-10) para o número de tigelas e os respectivos tamanhos naquela sala em específico (1-5). Depois, teremos de perguntar ao jogador quantos jogadores conseguiremos alimentar naquela sala. A resposta do jogador terá de ser verificada, temos de responder apropriadamente e confirmar se está correta (através do contador de respostas corretas). No final de cada interação teremos de aumentar o número da sala em 1.

[Passo 6]

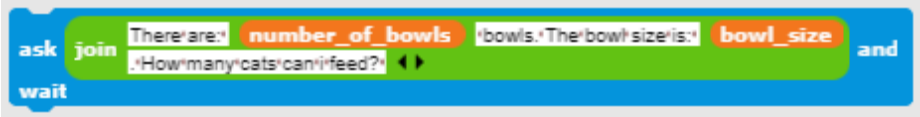
Para atribuir o número aleatórios de tigelas e os respectivos tamanhos, usaremos as [opções] *Variáveis/alteraçõe*s dos valores com *Operadores/um valor ao acaso entre [n] e [m]*.

[Passo 7]

Queremos perguntar aos jogadores pelo número de gatos que conseguimos alimentar dentro da [cadeia] de *Sensores/pergunta* e *bloco esperar*, caso contrário irá aparecer apenas por determinados segundos e depois será atualizada com outra nova frase. Desta forma, os jogadores acabam por esquecer rapidamente quantas tigelas/tamanhos existem na presente sala. De modo, é importante construir uma cadeia com origem em uma combinação de textos, referências e variáveis usando os blocos *Operadores/a junção de [Cadeia 1]*. Teremos de expandir este bloco para que a frase completa caiba.

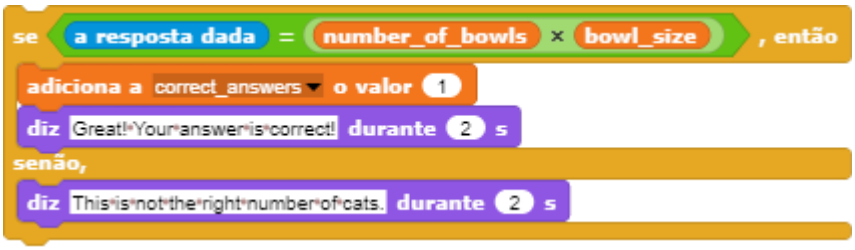
[Passo 8]

Temos de colocar esta longa frase no espaço dentro da [cadeia] e o *bloco esperar* para obtermos a resposta do jogador(a).



[Passo 9]

Quando o jogador responder, temos de verificar a resposta. Só existem duas situações possíveis, o jogador pode estar certo ou errado, então usaremos o bloco *Se-Então.* A resposta correcta é o valor correspondente da multiplicação das tigelas com os tamanhos das mesmas. Teremos de verificar se a resposta do jogador corresponde a esse número. Se a resposta estiver correta, aumentarmos para o 1 contador de respostas corretas. Se estiver incorreta, apenas apresentamos a resposta correta. Não devemos contabilizar as respostas erradas porque podemos calculá-las através do contador de respostas corretas.



[Passo 11]

Agora temos de selecionar um *loop*/ciclo. Como mencionado anteriormente, é melhor selecionar o *for loop* porque a variável que é usada para iteração é replicada para o contador das divisões do abrigo.

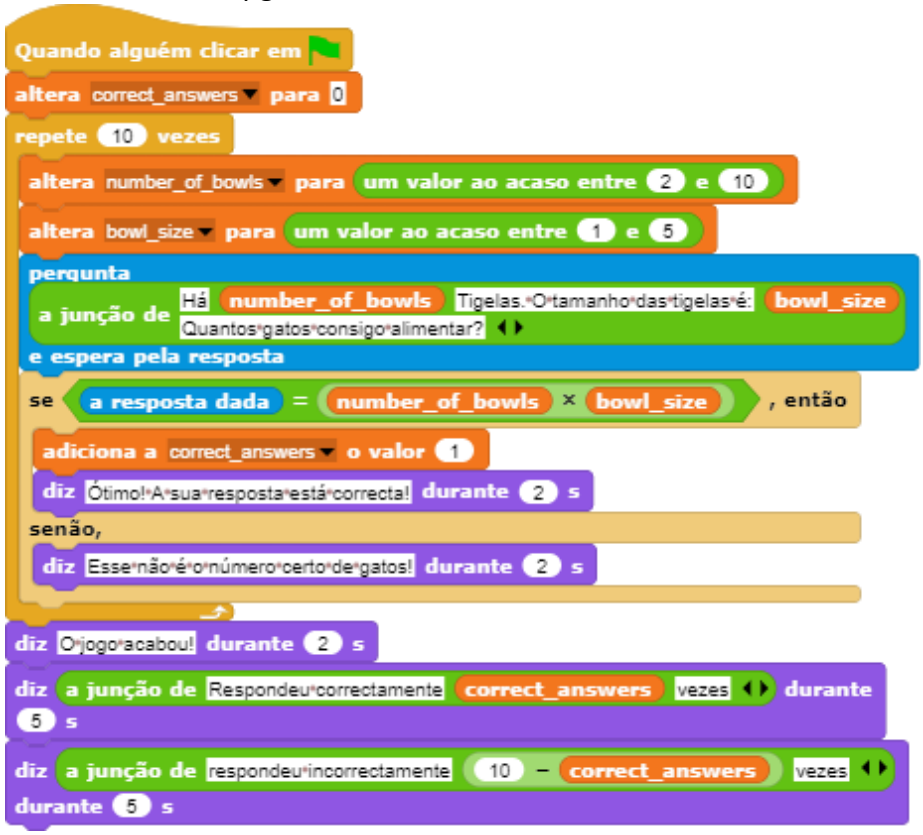
[Passo 12]

Quando o *loop*/ciclo parar, o jogo acaba. A partir daí fornecemos a informação com as conquistas do jogador. O Número de respostas correctas é armazenado no contador de respostas correctas; o número de respostas erradas pode ser calculado.

[Final code]

[Versão básica da atividade]

Para poupar tempo, podemos utilizar a versão básica do cenário. Nesta versão, todos os conceitos essenciais estão incluídos, outras funcionalidades descritas anteriormente podem ser atualizadas mais

	tarde através de upgrades. 
Instrumentos e recursos para o Professor	• Todas as atividades em Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_2 • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para alunos	• Template das atividades: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_template • Instruções para os alunos (C4G10_InstructionsForStudent.docx)

Cenário de Aprendizagem 11 - Adivinhar o número de gatos no abrigo.

Título do cenário de Aprendizagem	Adivinhar o número de gatos no abrigo
Experiência Prévia de programação	• condições (bloco *Se*) • imprimir o texto (bloco *dizer*)
Objetivos de Aprendizagem	Objetivos de Aprendizagem Gerais: • valores aleatórios, • atribuição de variáveis, • resposta do utilizador, • repetir até o loop/ciclo, • comparar operadores, • contador Objetivos de aprendizagem específicos orientados para um pensamento com base algorítmica: • Os alunos atribuem valores aleatórios para a variável, • Os alunos usam o bloco input para obter o número do jogador, • Os alunos usam repetir até ao loop, para perguntar repetidamente ao jogador para que ele introduza o número e realize o teste do valor, • Os alunos realizam o teste do valor com o *sentence if* e o comparador de operadores e dá a resposta apropriada, • Os alunos definem a condição de repetição do loop para verificar se o jogo realmente acabou, • Os alunos compreendem que não é necessário testar para verificar se o jogo acabou, porque já está implicitamente presente nas condições, • Os alunos devem implementar um contador para contar as tentativas do jogador e usar o valor final para distinguir os dois possíveis resultado.
Objetivo, Tarefa e Descrição breve das Atividades	**Breve Descrição**: Programar um jogo simples, em que logo no início é atribuído a uma variável um número aleatório entre 1 a 100. O jogador tentará adivinhar ao digitar os números. E obterão as respostas se o valor da resposta for : superior, inferior ou igual ao valor aleatório definido. **Tarefa:** Programar o abrigo da Martha para definir o número de gatos aleatoriamente, perguntar ao jogador o seu nome. De seguida, Marta terá de cumprimentar o jogador com o nome dele ou dela e depois perguntar repetidamente pelo número. Quando o jogador apresentar o número, ela deverá responder: 1) Se a tentativa for inferior ao número definido, ela diz: “O número de gatos é superior”, 2) Se a tentativa do jogador for superior ao número definido, ela deverá

	dizer: " O número de gatos é inferior", 3) Se a tentativa for correta, ela diz: "Excelente, acertaste no número de gatos". Programe o contador que irá contabilizar todas as tentativas do jogador. Quando o jogador adivinhar o número correcto, deverá verificar se o número de tentativas é inferior a 5. Nesse caso, o jogador recebe o gato, caso contrário não. **Objetivo: Os alunos serão ensinados a realizar as repetições até ao loop/ciclo e como definir as condições que serão capazes de rastrear implicitamente a condição que termina o jogo. Para além disso, aprenderão como usar variáveis em situações distintas: definir um valor aleatório, um contador ou a obter as respostas dos jogadores**
Duração da Atividade	45 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem ativa, colaborativa e com base na resolução de problemas
Formas de Ensino	Ensino presencial Trabalho individual/Trabalho em pares / Trabalho em equipa
Sumário da Lição	(Motivação-Introdução, Implementação, Reflexão e Avaliação) Marta, a guarda do abrigo de gatos, quer que adivinhem o número exacto de gatos que ela tem em seu abrigo. O número de gatos está entre 1-100. Quando o jogador(a) tentar adivinhar o número, ela responderá se o número é inferior, superior ou igual a resposta certa. Se o jogador(a) acertar o número em menos de 5 tentativas, receberá um gato, caso contrário, será convidado(a) a jogar novamente. [Passo 1] A primeira tarefa é fazer um cenário interessante para o jogo. Os alunos podem desenhar um por conta própria ou utilizar imagens com licença gratuita disponíveis online. Para poupar tempo, podemos preparar um cenário previamente.

[Passo 2]

Temos de selecionar um novo traje para a imagem (turtle sprite) que representará a guarda do abrigo.

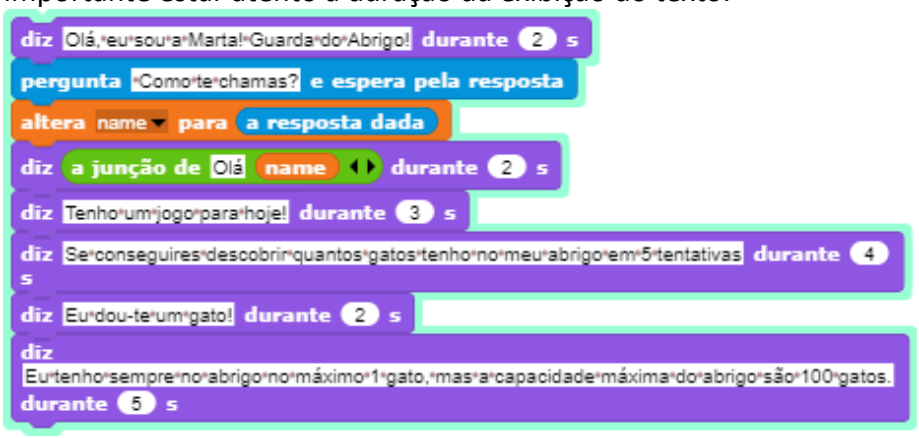
[Passo 3]

Nós discutimos com os alunos que este seria um jogo interessante para ser jogado mais de uma vez, se o número de gatos for definido de forma aleatória. De modo a tornar o número aleatório disponível para uma comparação das respostas, temos de os armazenar nas variáveis. Atualmente, as variáveis (assumimos que eles ainda não sabem os conceitos das listas) são a única forma de recordar um determinado valor no *snap* . Isto deve acontecer quando o programa inicia (*Evento/Quando alguém clicar em bandeira verde*).

[Passo 4]

A guarda do abrigo pergunta aos jogador(a) pelo seu nome para poder cumprimentar-lo(a). Isto é feito da seguinte forma, Sensores/pergunta[como te chamas] e esperar. A resposta do jogador(a) fica armazenado na built-in variável designada *resposta dada.* Para cumprimentá-la , temos de participar com algum cumprimento da [string] armazenada na variável *answer.* Isto é feito da seguinte forma*: Operadores/ bloco junção [string 1][string 2].* Para exibir o texto, usamos Aparência/diz [string] durante n segundos.

Também utilizamos esses blocos para escrever instruções de como deve-se jogar o jogo. Podemos também enfatizar que é muito importante estar atento a duração da exibição do texto.

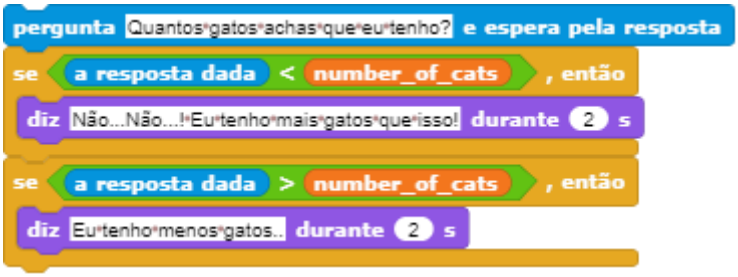


[Passo 5]

Discutimos com os alunos, a impossibilidade de prever quantas vezes os jogadores terão de tentar até chegarem ao valor certo. A pessoa pode ter muita sorte e descobrir de primeira ou talvez utilize a cinco tentativas para acertar ou até pode vir a precisar de mais, é dificil saber! E por isso é importante escolher o *loop* certo para esta tarefa. A guarda do abrigo tem de perguntar repetidamente pelo número e dar a resposta apropriada até que o jogador chegue no número correcto. O único *loop* que podemos utilizar para implementar a funcionalidade deseja é *repeated until[condition] loop*. A condição é relativamente fácil de ver, temos de realizar o *loop* até que o jogador responda, e fica armazenado na variável *built-in igual ao valor armazenado na variável cat_number* .

[Passo 6]

De seguida, temos de perguntar aos alunos quais são os comandos que irão no corpo do *loop/ciclo.* Qual é a atividade ou os comandos que serão repetidos até o jogador(a) acertar o número correto? Primeiramente, temos de perguntar ao jogador para inserir o número, depois temos de responder com base nesse valor.



[Passo 7]

A última questão a ser explicada ou discutida com os alunos é quando o *loop/ciclo* irá terminar e o que isso implica. Quando a resposta do jogador(a) for igual ao número certo de gatos, ambas as condições no corpo do *loop/ciclo* serão falsas, então o *loop* irá prosseguir para a próxima iteração, confirmando a condição do loop. E nesse caso, a condição será verdadeira, então o *loop* irá terminar e os comandos que seguem após o loop serão executados. Ou seja, quando o *loop* terminar sabemos que o jogador(a) acertou no número. A partir daí podemos responder de acordo.

	[Passo 9] Temos de criar uma nova variável que terá a função de de contador e o valor inicial será 0. Discutimos com os alunos a relevância do início da variável e a diferença entre definir o valor e aumentá-lo. Quando definimos o valor da variável, o valor anterior perde-se. E isso não está certo para o contador. Se aumentarmos o valor para algum número, nós adicionamos esse valor a qualquer valor que já estivesse presente anteriormente na variável. E neste caso, isso é exatamente o que queremos. Sempre que o jogador inserir um novo número queremos que o valor aumente em 1. [Passo 10] Depois de obter a resposta certa, temos de verificar se o valor da variável do contador para decidir se o jogador(a) receberá o gato ou não. Pois o *Snap* só tem operadores lógicos de menos (<) e não de menos ou igual, a condição para decidir se o jogador(a) fica com o gato ou não é *cat_counter* < 6. Este é também um bom exemplo para usar o bloco de condição *Se-Então* para diferenciar entre os dois casos. [Código Final]

[Versão Básica da Atividade]

Para poupar tempo, podemos utilizar a versão básica do cenário. Nesta versão, todos os conceitos essenciais estão já incluídos, as outras funcionalidades descritas anteriormentes podem ser usadas como atualizações mais tarde.

	
Instrumentos e recursos para o Professor	• Todas as atividades em Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para alunos	• Template da Atividade em Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter_template • Instrução para os alunos(C4G11_InstructionsForStudent.docx)

CENÁRIOS DE APRENDIZAGEM AVANÇADA

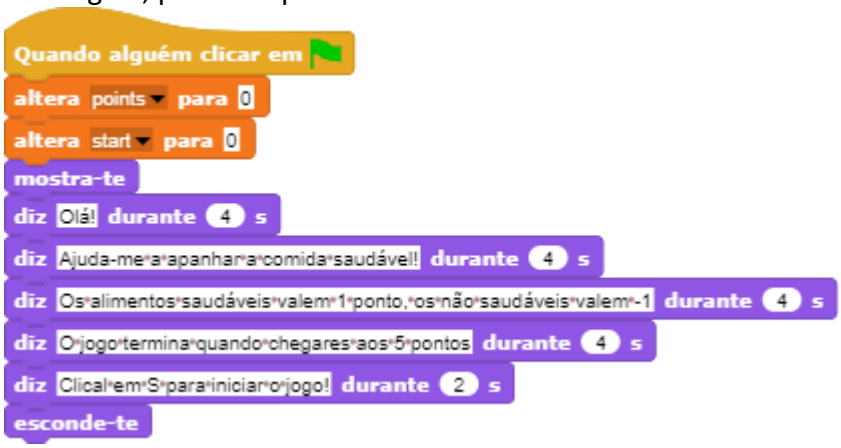
Cenário de Aprendizagem 12 - Apanhar comida saudável

Título do cenário de aprendizagem	Apanhar comida saudável
Experiência prévia de programação	Adicionar teste para *sprite* Mostrar e esconder *as imagens* Usar pontos de direção Usar valores ao acaso Uso de variáveis para contar os pontos Usar a repetição do loop/ciclo Usar o loop eterno Usar as condições
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Variáveis ● Condições ● Loop/ciclo ● Apontar em direção ● Valores ao acaso Objetivos de aprendizagem específicos, orientados para o pensamento algorítmico: ● Os alunos usam variáveis para impedir que o jogo comece antes que a rapariga acabe de falar (opcional) ● Os alunos usam *if statement* para verificar (com a ajuda da variável) se a comida pode começar a mover-se. ● Os alunos usam a repetição do *loop/ciclo* para o movimento da comida até que os pontos sejam inferiores a 5 ● Os alunos usam os pontos de direção 180 (para baixo) para a imagem mover-se para baixo. ● Os alunos escolhem aleatoriamente o número de passos. ● Os alunos escolhem aleatoriamente para mover em uma posição ao acaso ● Os alunos usam *um valor ao acaso* para mover da posição (opcional) de x (ao acaso) para y (fixo)
Objetivo, tarefa e breve descrição das atividades	**Breve Descrição:** A rapariga está a apanhar comida. Ela tem de ser cuidadosa, pois apenas alimentos saudáveis dão pontos ! **Tarefa:** Os alunos têm de programar duas imagens diferentes,

	uma rapariga que dá as instruções, diz o que fazer para iniciar o jogo e conta os pontos; e a comida que cai aleatoriamente da parte superior do ecrã. Para além disso, os alunos podem adicionar uma variável e *if statement* para prevenir que a comida se mova antes que a rapariga acabe de falar. **Objetivo: Os alunos devem aprender a como mover aleatoriamente por x passos e escolher a posição e como usar a variável as condições para prevenir outros eventos.**
Duração das atividades	45 minutos
Estratégias e métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design, resolução de problemas
Formas de ensino	Trabalho individual / Trabalho em pares
Sumário da Licao	(Motivação-Introdução, Implementação, Reflexão e Avaliação) A rapariga está a apanhar comida. Cada comida saudável traz 1 ponto, enquanto as comidas não saudáveis retiram 1 ponto. O jogo começa com algumas instruções, dadas pela rapariga. Depois disso, ela desaparece. Quando o jogador(a) alcançar os 5 pontos, a comida desaparece e a rapariga volta a aparecer.  [Passo 1] Esta atividade foi criada para ser desenvolvida individualmente ou em pares. A professora deverá dar algumas dicas, explicando as partes mais complexas e ajuda no que for necessário. Aos alunos é dado inicialmente: ● Cenário ● A imagem da rapariga

Os alunos escolhem o cenário e adicionam a imagem principal, como por exemplo, a rapariga. A rapariga dá algumas instruções no início e depois esconde-se. Como vimos nas atividades anteriores, é importante programar o bloco *mostrar* quando a bandeira for clicada (ao jogar novamente se a imagem permanecer escondida).

O código é, por exemplo:



Retornaremos a esta imagem mais tarde. Vamos escrever um código para a fruta agora.

[Passo 2]

Os alunos adicionam um novo *sprite* de um alimento saudável, por exemplo: uma maçã.

Primeiramente, eles devem programar o movimento da imagem (*sprite),* que é da parte superior para a parte inferior, e para isso selecionam os seguintes blocos:

Se não querem as as maçãs fiquem de cabeça para baixo, podem escolher a terceira opção *não roda* no bloco de pontos de direção.

Para tornar o jogo mais interessante, o número de passos pode ser escolhido ao acaso, então a velocidade pode não ser sempre a

mesma. Por exemplo:

Neste caso, o aluno pode utilizar o bloco *está a tocar na borda* juntamente com a condição *if*. Se a maçã tocar no limite, será movida para uma posição ao acaso. Os Blocos de movimento nos disponibilizam o seguinte bloco:

Este comando escolherá ao acaso qualquer coordenada e poderá aparecer em qualquer outro lugar do ecrã (repare nos pontos vermelhos da imagem).

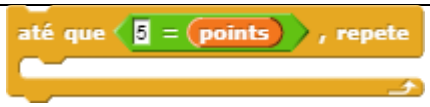
Se queremos que a maçã apareça sempre na parte superior do ecrã, o valor de y pode ser definido de forma fixa, e apenas o valor de x será escolhido aleatoriamente. Com o código seguinte a maça irá aparecer sempre no topo do ecrã (repare nos pontos vermelhos da imagem)

[Passo 3]

Agora os alunos conseguem a variável dos *pontos*, que será usada para contar. A pontuação deve de estar definida com o valor 0 no início (na imagem da rapariga).

[Passo 4]

Se queremos que a maçã movimente-se constantemente, precisamos de um *loop/ciclo.* Os alunos podem usar o loop *repetir até* e definir a condição. Por exemplo, queremos que o jogo acabe quando os 5 pontos são alcançados. Então a condição será pontos = 5 e o *loop* irá repetir até que a condição seja falsa.
Quando a condição for verdadeira, ou seja, quando a pontuação for equivalente a 5, o *loop* para.

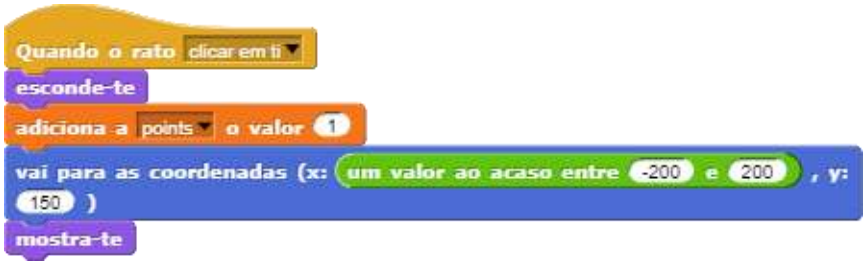


[Passo 5]

Não queremos que a maçã apareça logo no início, apenas depois das instruções serem dadas pela rapariga.Os alunos podem programar para que a maçã apareça quando *key é selecionada.* Mas para isso, eles têm de adicionar o bloco *mostrar* antes da repetição do *loop/ciclo* e *desaparecer* depois disso. Por agora o código completo está desta forma:

[Passo 6]

O que acontece quando a maçã é selecionada (ou clicada pelo rato) ? A maçã tem de esconder-se, contar como pontuação,mudar de posição e aparecer novamente. Os pontos mudam de 1 em 1 e para a posição o código pode ser o mesmo que o anterior.



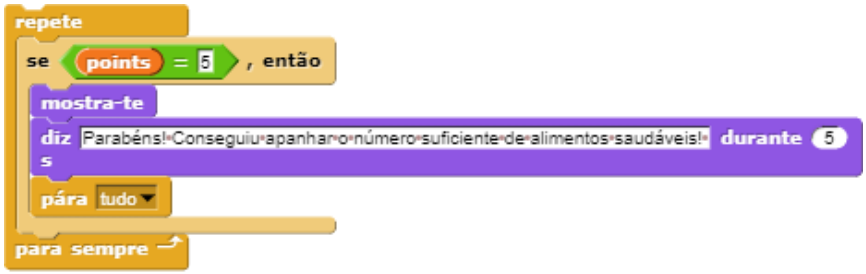
[Passo 7]

Vamos voltar a rapariga!

A rapariga deve aparecer e dizer, por exemplo "*Parabéns!*"

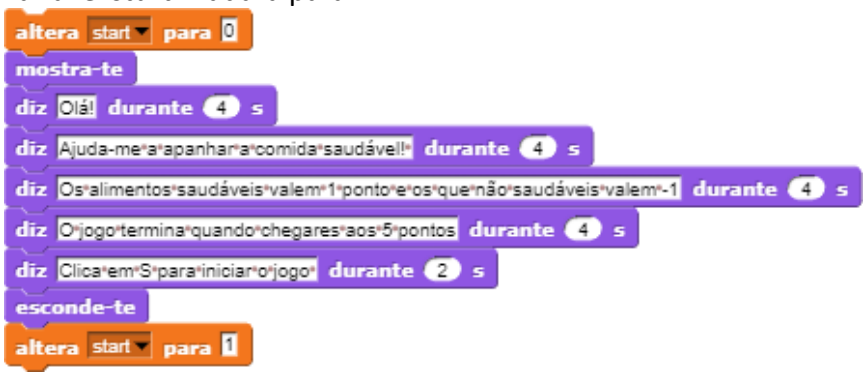
Precisaremos de um *loop* eterno, que irá verificar se o jogador(a) chegou aos cinco pontos. Se sim, a rapariga irá aparecer e dizer algo. Depois disso, adicionaremos o bloco *parar tudo*. E deixar os

alunos descobrirem sozinhos o que isso significa (sem o sinal de parar a rapariga estará presente no ecrã com a frase *Parabéns* por um tempo indefinido).

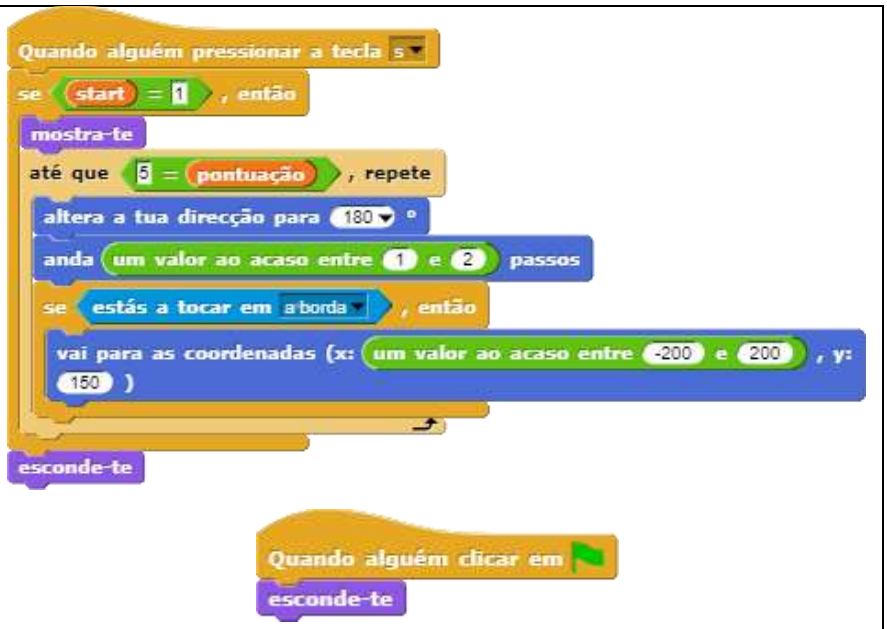


[Passo 8]

Quando o jogo for jogado novamente, os alunos já terão conhecimento das instruções (do [Passo 1]) e vão desejar saltar essa informação. Para isso podem clicar o "S" antes, assim o jogo irá começar, mas a rapariga continuará falando. Para prevenir isso podemos criar outra variável (chamada *start*), que deverá estar definida como 0 inicialmente. Assim, depois das instruções, a variável *start* mudará para 1.



Agora, temos de programar para que a maçã só apareça quando a variável *start* seja igual a 1, e os alunos farão isso com *if statement.* Com isso, os alunos não serão capazes de executar o jogo antes que a rapariga pare de falar.
Outra coisa pode acontecer quando jogamos novamente. Se paramos o jogo quando temos, por exemplo, 3 pontos, a maçã não irá desaparecer. Neste caso, quando começarmos o jogo novamente, a maçã estará visível antes que a rapariga acabe de falar. Como não queremos isso, adicionamos o código para que a maçã se esconda no início do jogo.
Agora o código da maçã está da seguinte forma:

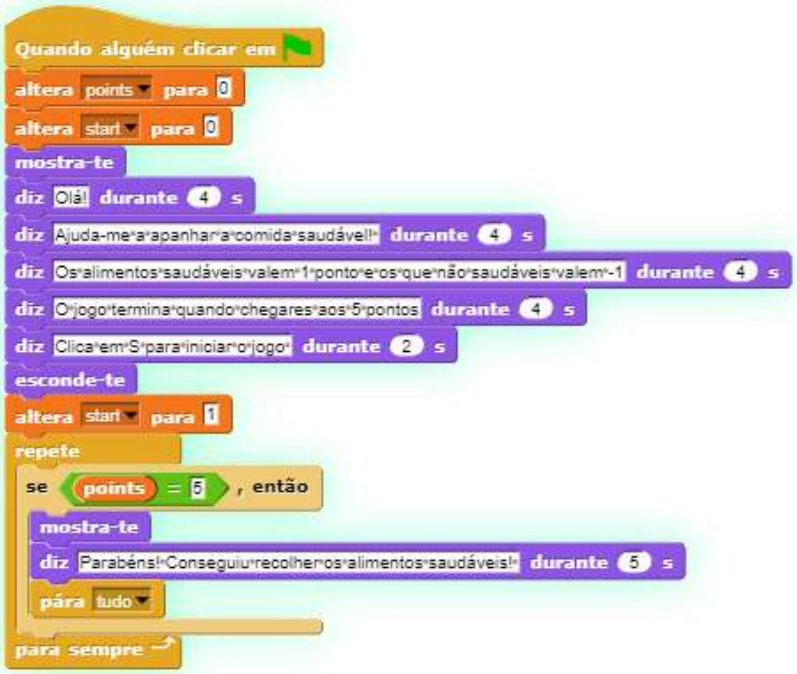


[Passo 9]

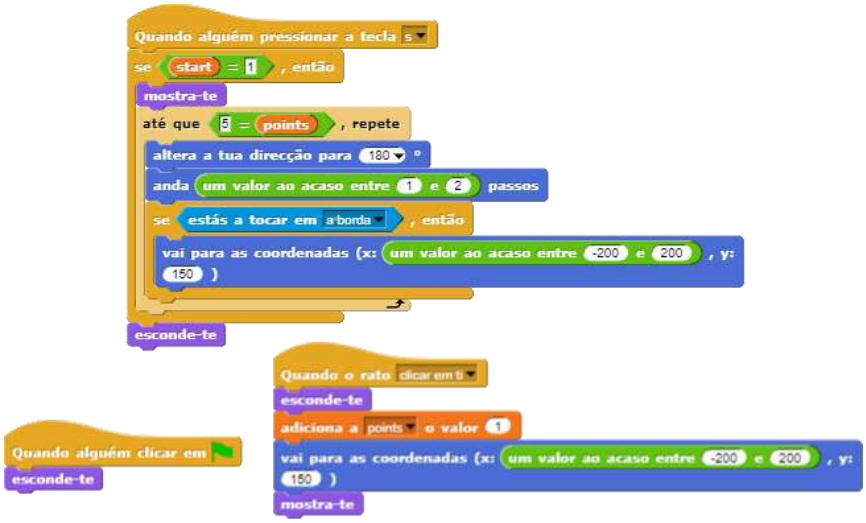
Os alunos agora podem duplicar a imagem da maçã muitas vezes e mudar o traje (se quiserem).
O código será o mesmo.
A única diferença é com a comida que não é saudável, onde eles perdem 1 ponto ao selecioná-la.

[Código Final]

Rapariga



Maçã

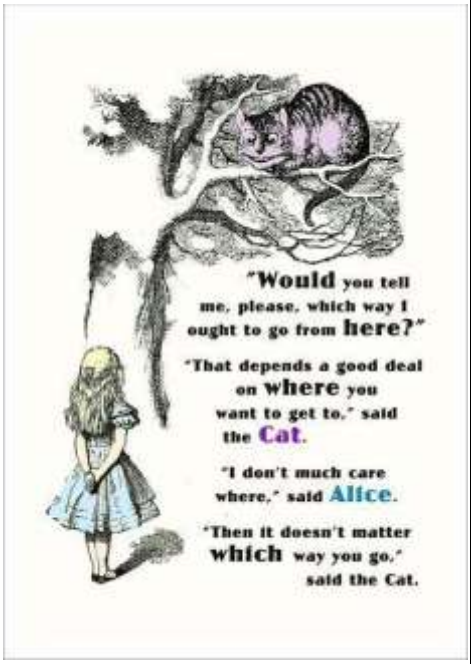


[Tarefas Adicionais]
Os alunos podem adicionar tarefas adicionais de acordo com a sua vontade, e podem seguir as seguintes tarefas:

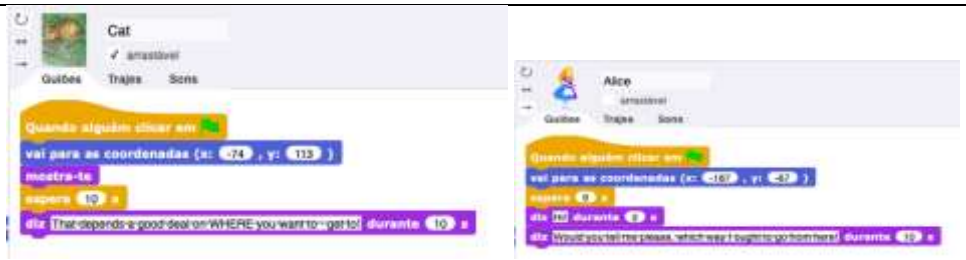
- Mudar o jogo para que a imagem da tigela apanhe o alimento.
- Adicionar uma nova imagem (uma tigela). É possível desenhar, encontrar online ou anexar uma fotografia para a imagem da tigela.
- Definir a posição de início (por exemplo: na parte superior

	do ecrã) e escrever o código para o movimento da tigela (esquerda e direita, e se quiser, cima e baixo). As imagens dos alimentos têm de desaparecer e aparecer em outra localização aleatória quando encostam na tigela (e não mais com o clique do rato no alimento, como anteriormente). ● Mudar as regras – Deixe que o jogo acabe quando o jogador chegar aos 20 pontos (o jogador(a) vence o jogo) ou quando ele apanhar 3 alimentos que não são saudáveis (jogador(a) perde). ● Adicionar mais imagens de alimento para tornar o jogo mais interessante. ● Mudar o traje da tigela quando o jogador atinge uma determinada pontuação, por exemplo: 5,10,15 pontos.
Instrumentos e recursos para o Professor	● Todas as atividades em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food ● Todas as atividades em Snap! Com tarefas adicionais (soluções possíveis): https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food%20%2B%20Add.%20Task ● Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para alunos	● Atividade pré-construída em Snap!: https://snap.berkeley.edu/project?user=mateja&project=C4G12_Catching%20healthy%20food%20-%20Part ● Instruções para o aluno (C4G12_InstructionsForStudent.docx) ● Imagens: bowl1.png, bowl2.png, bowl3.png, bowl4.png

Learning Scenario 13 - Storytelling

Título do cenário de Aprendizagem	Storytelling
Experiência Prévia de programação	Mostrar e esconder a imagem (sprite) Utilizar condições Utilizar *diz* (dos looks group) Utilizar espere por … segundos
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● Mover-se e mudar de tamanho ● Difunde a mensagem ● Compor a estrutura do storytelling ● Mudar o cenário das cenas ● Objetivos de aprendizagem orientados pelo pensamento algorítmico: ● Os estudantes planeiam os diálogos e atividades das imagens dentro da história ● Os estudantes usam *difunde a mensagem* para os diálogos entre imagens ● Os estudantes usam mover e mudar de tamanho para as imagens ● Os estudantes usam mostrar e esconder imagens ● Os estudantes refatoram e extendem o código dos sprites (imagens)
Objetivo, Tarefa e uma breve descrição das atividades	Breve descrição: O coelho conta a história da Alice no país das Maravilhas. Ele inicia o storytelling com diversas frases contrárias ao cenário rotulado em Alice no país das maravilhas. A história da Alice começa na floresta. Ela anda e questiona-se "Onde estou?" / Para ver a Alice mover-se para longe, o seu tamanho é reduzido gradualmente junto ao movimento. Ela chega na encruzilhada e vê o Gato Cheshire numa árvore. Inicia-se uma conversa entre a Alice e o Gato Cheshire. A conversa está representada na imagem: **Tarefas:** Os estudantes devem 

	experimentar através de um breve exemplo da história do encontro entre a Alice e o Gato, baseando a sincronização do diálogo por um bloco de pausa. Depois eles revêem uma segunda versão da história usando *difunde a mensagem*. Os comandos de mensagens são inseridos. Os estudantes completam o código das personagens de acordo com o texto da imagem. A tarefa é complicada ao introduzir a mudança da decoração do cenário por *difunde a mensagem* e *mover* a Alice pela floresta antes dela encontrar o gato. **Objetivo: Os estudantes aprenderão a planear um storytelling, como utilizar *difunde a mensagem* para sincronização das atividades das imagens (sprites) e mudanças do stage**
Duração das atividades	90 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design e resolução de problemas
Formas de Ensino	Trabalho individual/Trabalho a pares / Discussões presenciais
Sumário da Lição	(Motivação-Introdução, implementação, Reflexão e avaliação) 1. O professor(a) discute com os alunos a história da Alice no país das Maravilhas e mostra a imagem da Alice encontrando o Gato Cheshire. O professor ou professora explica que a história da Alice pode ser recriada através da utilização de codificação. Os estudantes são encarregados de iniciar o projecto e ver os códigos das imagens (sprites). https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 *Discussão: Quem começa a falar primeiro? Quando Alice é envolvida e quando - o Gato? Porque não há sincronização no diálogo dos personagens? A resposta está na inexatidão do cálculo do tempo em que cada personagem "fala" e "a falta de intervalo para esperar por um personagem terminar sua resposta".* Os códigos são comentados e a tabela é completada:



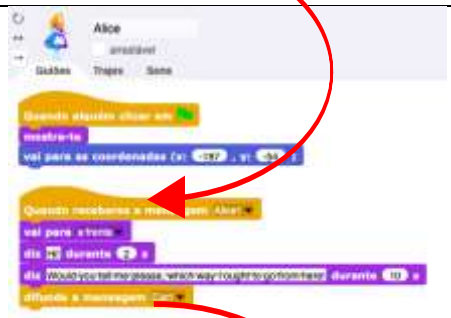
Sprite	Atividade	Começar no início	Momento final	Duração
Coelho	Diz: Olá! Já ouviste falar da Alice e das suas aventuras no país das maravilhas? Agora vamos ver uma das suas histórias.	0	14	14
Alice	Diz: *Poderia dizer-me, por favor, para qual caminho eu devo seguir a partir daqui*?	9	21	12
Gato	Diz: Isso depende de ONDE queres chegar!”	10	20	10

A conclusão é que sincronizar com o bloco espere por … segundos pode causar erros no comportamento dos personagens dentro do storytelling.

2. O professor é encarregado em iniciar e rever o projecto de código https://snap.berkeley.edu/project?user=ddureva&project=Alice_2

Quais são os comandos não familiares até agora?

Os códigos da Alice_1 e Alice_2 sao comparados.

Alice_1	Alice_2
	
	
	

3.. Blocos para difundir a mensagem são introduzidos:

É discutido que as mensagens difundidas têm como alvo todas as personagens, mas só podem ser recebidas por algumas personagens. Difunde a mensagem … e o bloco de espera requer, a todos os personagens que receberam, performar suas ações e depois as ações do sprite (imagens) que enviou a mensagem continua.

O professor demonstra como nomear uma mensagem difundida e como é usada no evento Quando eu receber…

1. 

2. 

3. Introduzir um nome . OK

Utilize em um evento:

1. 

,

2. A mensagem que deve ser recebida pelo sprite deve ser selecionada na lista.

3. O grupo discute como completar a história na imagem. Como nomear as mensagens, por exemplo: A mensagem do Gato para Alice para ser Alice2 e de Alice para o Gato - Gato1.

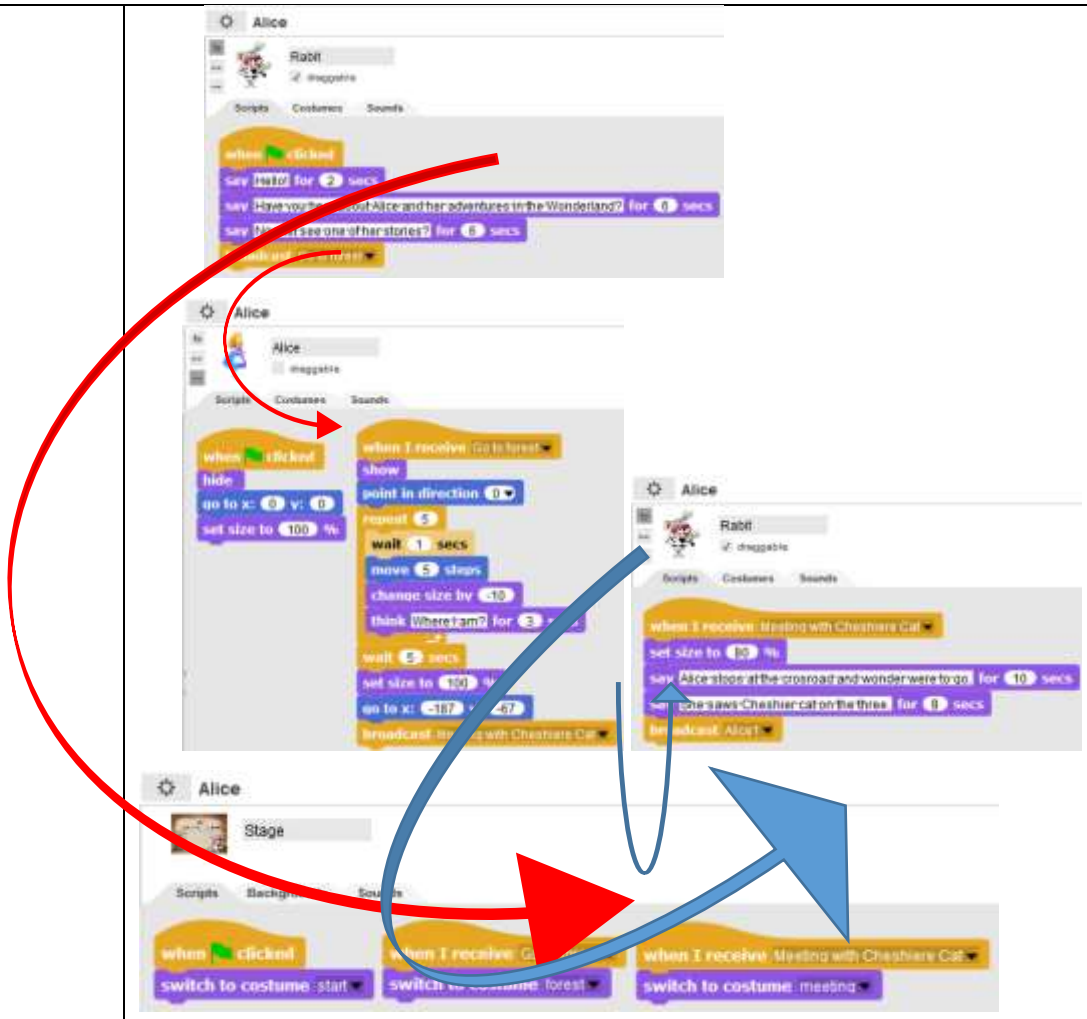
4.Os estudantes completam a história em pares.

5. O professor comenta que contar histórias normalmente requer uma mudança nos trajes no stage. *"Vamos fazer a história da Alice mais completa ao iniciar a história do Coelho contrariamente ao contexto introdutório, movendo a ação para dentro da floresta onde a Alice está a andar e pergunta-se: "Onde estou?" E o seu tamanho diminui gradualmente enquanto ela move-se para longe. Depois, ela encontra-se em uma encruzilhada e vê o Gato Cheshire. A conversa entre os dois é iniciada.*

6. O professor demonstra o projeto.

https://snap.berkeley.edu/project?user=ddureva&project=Alice

Mudanças nas cenas e nas ações dos personagens são comentadas.
"Quando é que uma cena muda? Quando é que Alice aparece e quais são as suas ações? Quando é que o Gato aparece e quais são as suas ações?"
As cenas no projeto Alice_2 são discutidas. Há 3 cenas, uma já usada. Que cena usar para iniciar? O que deve ser feito para prevenir os sprites da Alice e o Gato de serem mostrados no início? Como mudar a decoração do palco? O *difundE mensagem* pode ser utilizado para mudar a definição do palco pelo Coelho após introduzir suas palavras introdutórias. Alice aparece quando a cena muda com a mensagem *Vai para a floresta.*



Quando a Alice está no caminho para a floresta, ela anda e pergunta-se, então para com maior concordância com a realidade, o tamanho dela diminui -10%. Isso repete-se 5 vezes usando o *repeat loop.*

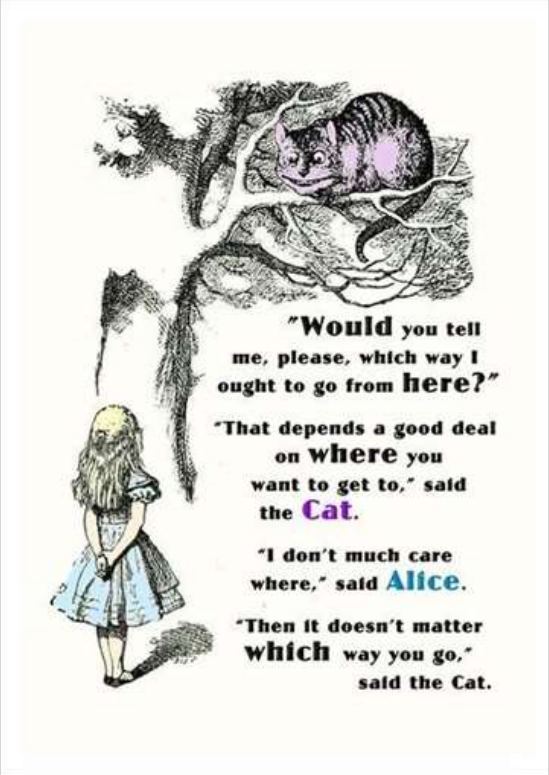
Quando ela atinge a junção, a cena muda com a mensagem "A encontrar o Gato Cheshire". Essa mensagem é recebida ao mesmo tempo pelo Coelho, que tem seu tamanho reduzido a 80% e ele continua a contar a história com seu tamanho reduzido.

Nesta fase, a imagem do gato não é mostrada porque está presente com parte da decoração.

Aparece na mensagem Gato1. O professor pode explicar que o gato foi cortado da decoração utilizando um editor gráfico externo. (Infelizmente, Snap! não provém muitas capacidades de edição gráfica, assim como Scratch 3.0).

Depois da aparição da mensagem do Coelho, a história da Alice1 continua assim como foi feito no projeto Alice2.

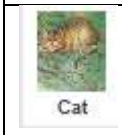
3. O professor comenta que para contar uma história deve-se ter um enredo. Uma table adicional pode ser usada para descrever o cenário da

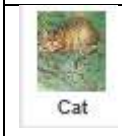
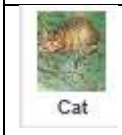
	história. (Appendix 1) Ao critério do professor a table finalizada pode ser oferecida completa ou parcialmente completa e os estudantes, guiados pela ilustração, podem completá-la. 4. Os estudantes são encarregados de descrever os cenários examinados e completar a história do projeto Alice_2 em pares.
Referências/ Materiais para os estudantes	Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Alice
Referências/ Materiais para os estudantes	 ● https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 ● https://snap.berkeley.edu/project?user=ddureva&project=Alice_2 ● Instructions for student (C4G13_InstructionsForStudent.docx)

Appendix 1. Enredo da história/ Cenários

Nome	Design	Ações	Notas
1. Início		A história começa com a cena (Quando a bandeira verde é clicada)	Em oposição a esse cenário o Coelho introduz a história.
2. Floresta		O cenário aparece quando o Coelho se refere à sua introdução (Uma mensagem de *Vai para a floresta* foi enviada)	Em oposição a esse cenário, Alice aparece posicionada no centro do stage. Ela começa a mover-se e a perguntar-se "Onde eu estou?" . O sprite gradualmente tem seu tamanho reduzido por 10% 5 vezes. Quando chega ao fim do caminho (na encruzilhada), a cena muda para Encontro. (Alice - manda difunde a mensagem- Encontro com o Gato *Cheshire*).
3.Encontro		Aparece quando a mensagem da Alice: *Encontro com o Gato Cheshire* é recebida.	Aqui Alice e o Gato são parte do cenário. Para utilizar a imagem da Alice, antes da mensagem, ela é posicionada para que ela cubra a própria imagem na decoração. A imagem do Gato aparece após alguns momentos. Enquanto o cenário muda o Coelho continua a contar a história. Mais tarde a conversa é entre Alice e o Gato Cheshire.

Imagens (Sprites)

Imagens (Sprite)	Ações	Cenário do palco
	No início: Diz: Olá! (Por 2 segundos) Diz: Já ouviste falar da Alice e das suas aventuras no país das Maravilhas? (Por 6 segundos) Diz: Agora vamos ver uma das suas histórias! (Por 6 segundos) Envia a mensagem *Vai para a floresta*.	
	No início: Esconde do palco; a posição no centro do palco e tamanho 100%, pronto para ser exposto contra o novo cenário.	
	No início Esconde do palco; posicionado em x: -74, y: 133 (As posições são pré-determinadas depois do Gato Cheshire ser posicionado no palco do Encontro.	
	Recebe a mensagem *Vá para a floresta*: A imagem aparece no palco. Repetido 5 vezes: esperando por 1 segundo; movendo 5 passos; redução de tamanho (mudança de -10); indagamento: *Onde eu estou?* Preparando para a próxima decoração: esperando 2 segundos; restaurando o tamanho do Sprite (mudança de 100%) e posicionando x em: -187, y: -67 Enviar mensagem: *A encontrar com o Gato Cheshire*.	
	Sem ação. Apenas torna-se visível a decoração anterior.	
	Recebe a mensagem*: A encontrar com o Gato Cheshire.* Tamanho muda para 80% *Ele diz: "Alice para na encruzilhada e pergunta-se para onde ir. (por 10 segundos).* *Ele diz, "Ela viu o Gato Cheshire na árvore. " (por 8 segundos)* *Envia uma mensagem Alice1*	
	Recebe a mansagem *Alice1*. Move-se para frente (Isso é necessário porque o Gato aparece depois dela, que previne que as linhas da Alice apareçam na caixa de diálogo se ela não estiver na camada da frente). Ela diz: *"Olá!"* (por 2 segundos). Ela diz *"Poderia dizer-me, por favor, para qual caminho eu devo seguir a partir daqui?" (*por 10 segundos).	

	Enviar uma mensagem de transmissão para o Gato: Gato1.	
	Recebe a mensagem do Gato1. O Sprite aparece no palco. Ele diz: “Isso depende de ONDE queres chegar!” (por 10 segundos). Envia uma mensagem *Alice2*.	
	Recebe a mensagem Alice2. Diz: …… Envia a mensagem Gato2.	
	Recebe a mensagem Gato2. Diz: …… Envia a mensagem Coelho1.	
	Recebe a mensagem Coelho1. Diz: “Qual é a moral da história?” (por 8 segundos) Diz: “Para saber que caminho seguir, deve-se primeiramente determinar o ponto de chegada”	

Cenário de Aprendizagem 14 - Desenhar

Título do cenário de aprendizagem	Desenhar
Experiência prévia de programação	Adicionar sprite (imagem) Utilizar ponto de direção Utilizar variáveis para contar ponto Utilizar loop repeat Utilizar condicionais
Objetivos de aprendizagem	Objetivos Gerais de aprendizagem: ● Variáveis ● Condições ● *Loop* (ciclos) ● Ponto de direção ● Operadores Objetivos de aprendizagem específico orientados por raciocínio algorítmico: ● Estudantes usam caneta para desenhar ● Estudantes usam loops para desenhar ● Estudantes mudam o valor da variável enquanto desenham ● Estudantes usam ponto em direção para desenhar objetos no palco ● Estudantes usam transmissão para controlar o sprite ● Estudantes usam condicionais para mudar o stage ● Estudantes usam operador > para mudar o stage
Objetivo, Tarefa e uma breve descrição das atividades	Breve descrição: O clima mudou muito, o ar está extremamente poluído devido às indústrias. As árvores precisam ser plantadas a fim de melhorar a qualidade do ar! Tarefas: Para melhorar a qualidade do ar, estudantes precisam programar um sprite para desenhar 2 diferentes tipos de árvores – pinheiro e carvalho, e botões que simbolizem os diferentes tipos de árvores. Quando o botão é clicado, um tipo específico de árvore é desenhado. **Objetivo: Estudantes aprenderão a desenhar no Snap!, para mudar de cor e a grossura da caneta, e como usar as variáveis e as condições que causam um evento.**
Duração das atividades	45 minutos

Estratégias e métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design, resolução de problemas
Formas de ensino	Trabalho individual / Trabalho em pares
Sumário da lição	(Motivação - Introdução, Implementação, Reflexão e Avaliação) No início do jogo, uma indústria que causa mudanças climáticas e a variável que mostra a qualidade do ar aparece no stage. É preciso plantar árvores para melhorar a qualidade do ar. Dois diferentes tipos de árvores podem ser desenhadas, pinheiro e carvalho. Quando um pinheiro é desenhado, o ar melhora em 3, e ao desenhar um carvalho o ar melhora em 2 unidades. Quando a qualidade do ar atinge 10 unidades, o cenário do stage muda para um campo. [Passo 1] Estudantes precisam abrir o programa Improve *the Climate* que contém modelos de cenários (indústria e relva) e sprites (um lápis, um pinheiro, um carvalho e um sprite chamado *clear*). Também, adicione um novo sprite – lápis (“lápis” das imagens oferecidas). Porque o sprite (imagem) é muito grande, deve-se ser reduzido a 50%. A posição inicial do lápis (coordenadas) deve-se ser especificada, por exemplo: X= -10, y = -10.   [Passo 2] A imagem do lápis (sprite) deve receber mensagens “pinheiro” e “carvalho” e desenhar as árvores apropriadas em resposta a mensagem.

Primeiramente, marque o lápis e adicione o código que capacitará desenhar o pinheiro usando uma caneta quando o sprite receber a mensagem “pinheiro”.

Um ponto em direção deve ser definido em 90º para desenhar a copa da árvore no formato de um triângulo, e as suas cores devem ser definidas.

Para desenhar a copa do pinheiro, mova a imagem (sprite) 40 passos e vire 120 graus a esquerda.

Esse movimento deve-se repetir 3 vezes.

Após da copa da árvore de pinheiro, o tronco também deve ser desenhado. Para que o tronco esteja na posição correta, mova 22 passos. Após isso, é importante definir a cor da caneta para castanho.

Vire 90 graus para a direita, e depois mova 10 passos.

Esse movimento deve ser repetido 3 vezes.

No final, é necessário levantar a caneta para que a imagem (sprite) não deixe um traço durante o próximo movimento. Para além disso, a caneta deve ser movida para uma posição aleatória.

[Passo 3]

Assim como anteriormente, é necessário adicionar o código do lápis para desenhar os carvalhos. O carvalho deve ser desenhado quando a imagem (sprite) receber a mensagem "carvalho". Um ponto em direção deve ser definido em 90º para manter a copa da árvore redonda, a caneta deve estar abaixo e a cor deve ser escolhida.

Para desenhar a copa do pinheiro, mova o sprite 1 passo e vire 3 graus a esquerda após cada passo.

Esse movimento deve ser repetido 120 vezes.

Uma vez em que a copa de árvore está finalizada, o tronco deve ser desenhado. O lápis deve ser movido para o centro para desenhar o círculo, em -3 passos, e a cor da caneta deve ser trocada para castanho.

Para desenhar o tronco, a imagem (sprite) deve estar virado 90 graus a direita e movido 10 passos.

Essa parte é repetida 3 vezes.

Quando o desenho está finalizado, é necessário levantar a caneta para que se desenhe uma linha enquanto se move a imagem (sprite).

Após o carvalho estar desenhado, a caneta deve ser movida para uma posição aleatória.

[Passo 4]

Em seguida, os alunos devem adicionar o código que faz com que todas as árvores sejam excluídas quando clica-se em *limpar sprite.* Quando o limpar sprite é clicado com com o rato, é transmitido uma mensagem

para limpar todas as árvores. Quando o lápis recebe a mensagem, todas as árvores desenhadas são eliminadas.

[Passo 5]

Crie uma nova variável “ar limpo” para mostrar a qualidade atual do ar. Defina o valor inicial em 0 e mostre a variável no palco.

Todas as vezes em que um pinheiro é desenhado o ar melhora 2 unidades, então adicione o bloco ao pinheiro sprite que terá o valor da variável “ar limpo” mudado em 2 unidades cada vez em que se clica no pinheiro.

Cada vez em que um carvalho é desenhado o ar melhora 3 unidades, então adicione o bloco a a imagem do carvalho que terá o valor da variável “ar limpo” mudado em 3 unidades cada vez em que se clica no carvalho.

[Passo 6]

Quando a variável “ar limpo” atingir 10, o stage deve mudar para relva. Portanto, através do materiais descarregados adicione um novo cenário “relva” para o palco (cenário é dos materiais descarregados).

Adicione um bloco de início „Quando“ da paleta de „Controle“ ao lápis.

Então, adicione operador >.

Defina para que o sprite transmita a mensagem “relva” quando a variável “ar limpo” estiver maior que 10.

Adicione o código ao palco para mudar o traje para “relva” quando a mensagem “relva” for recebida.

[[Tarefa Adicional]

É possível fazer um upgrade no jogo ao adicionar animais que aparecem

quando o ar não está mais poluído.

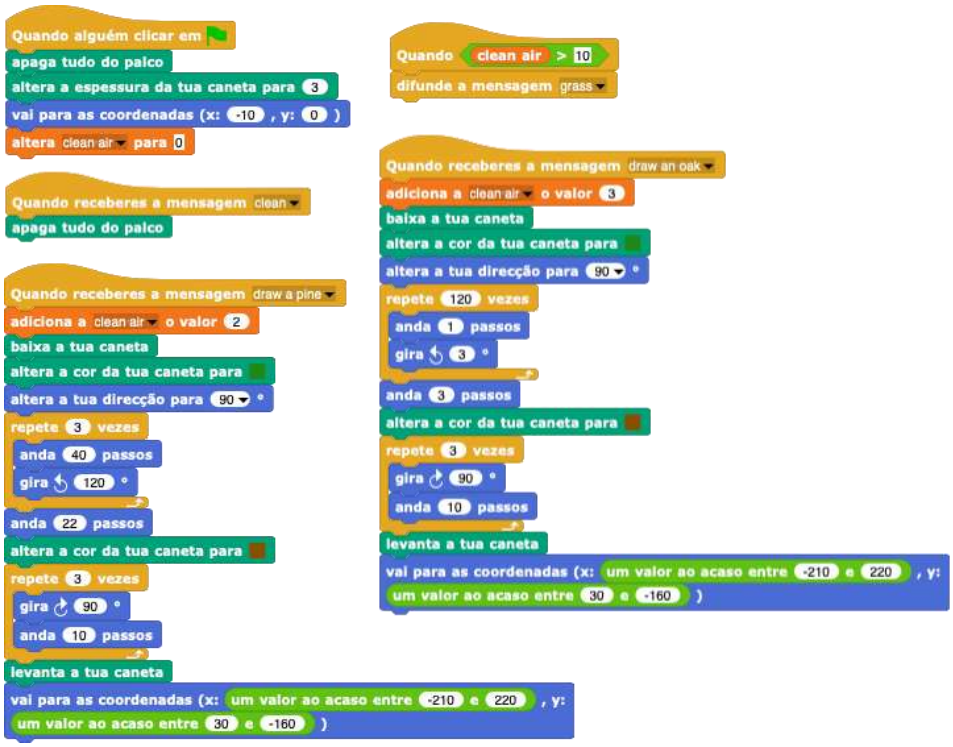
[Código final]

Pinheiro

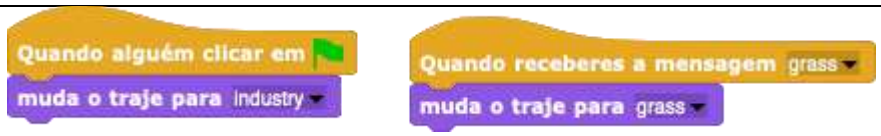
Carvalho

X

Lápis



Stage

	
Referências/ Materiais para os estudantes	Snap! projeto "Desenhar": https://snap.berkeley.edu/project?user=tadeja&project=Improve%20the%20climate (9.1.2020)
Referências/ Materiais para os estudantes	• Programar a Línguagem no Snap!: https://snap.berkeley.edu/ (9.1.2020) • Instruções para o aluno (C4G14_InstructionsForStudent.docx)

Cenário de Aprendizagem 15 - Apanhar o rato

Cenário de Aprendizagem Título	Apanhar o Rato
Experiência Prévia de Programação	● O aluno pode adicionar um novo cenário. ● O aluno pode adicionar um novo sprite. ● O aluno pode adicionar um novo som. ● O aluno sabe como colocar o sprite a dizer algo. ● O aluno sabe como alterar a fantasia de sprite para fazer uma animação ● O aluno sabe movimentar o objeto com as setas, usando eventos e tendo em consideração as restrições. ● O aluno sabe diferenciar dois estados diferentes e sabe como expressá-los com expressões lógicas. ● O aluno sabe como usar as condições.
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● *Loop* infinito; ● números aleatórios; ● contador; ● cronómetro. Objetivos de aprendizagem específicos orientados no pensamento algorítmico: ● O aluno usa *loop* infinito para mover os *sprites*; ● O aluno usa números aleatórios para determinar a posição do *sprite*, mover o *sprite* para etapas aleatórias e girar o *sprite* para graus aleatórios. ● O aluno implementa o contador para contar a captura dos ratos e usa o valor final para avaliar o sucesso do jogador. ● O alunos usa o cronômetro para determinar o fim do jogo.
Objetivos, Tarefas e uma Breve Descrição das Atividades	**Breve Descrição**: Programar um jogo em que o jogador (o gato) tem como objetivo apanhar o rato. **Tarefas**: Programar a actividade na qual o gato vai apanhar o rato. O gato vai ser movido pelo jogador com as setas e o rato vai-se mover aleatoriamente. Quando o gato toca no rato, o rato vai esconder-se e aparecerá numa localização aleatória. Existe também o contador que vai contar o número de vezes que o gato apanha o rato. Também é necessário o cronómetro para terminar o jogo. Depois da atividade, a rapariga tem que resumir o quão bem sucedido o jogador foi, para isso ela conta quantas vezes o jogador apanhou o rato. **Objetivo**: **O aluno será introduzido ao conceito de atribuição de valores aleatórios de variáveis múltiplas. Eles aprenderão a usar o bloco "Operadores/escolher ao acaso *[x] a [y]*"**
Duração das Atividades	45 minutos

Estratégia e Métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem colaborativa, solução de problemas e aprendizagem através de game-design .
Formas de Ensino	Ensino presencial Trabalhar em pares/ trabalho de grupo
Sumário da lição	(Motivação -Introdução, Implementação, Reflexão e avaliação) **Motivação-Introdução** Motivar os alunos através da demonstração do jogo. Debater com eles como é que devem começar a programar este jogo. Juntamente, com o alunos, determinar as sequências de passos, por exemplo : 1. Escolher o cenário e adicionar *sprites*; 2. Programar o gato para se mover com as setas; 3. Programar o rato para se mover aleatoriamente; 4. Programar o rato para se esconder(e aparecer numa localização aleatória) quando toca no gato; 5. Programar o contador ; 6. Adicionar o cronómetro e determinar o fim do jogo; 7. Adicionar a rapariga e programá-la para resumir quão bem sucedido o jogador foi; 8. Programar a rapariga para saltar quando ela toca no rato; 9. Adicionar o som do gato/rato 10. etc. 11. Os alunos podem ajudar com os passos ou criar as suas próprias regras do jogo(mas têm de seguir passos arrojados).  Introduzimos o operador para a atribuição de valores ao acaso.  Os alunos irão programar as seguintes tarefas em pares/grupos com a ajuda do professor.

Implementação

[Passo 1]

O primeiro passo é determinar o fundo do jogo. Os alunos pesquisam por imagens grátis online. Depois, adicionam as novas imagens (*sprites)* - o gato e o rato.

[Passo 2]

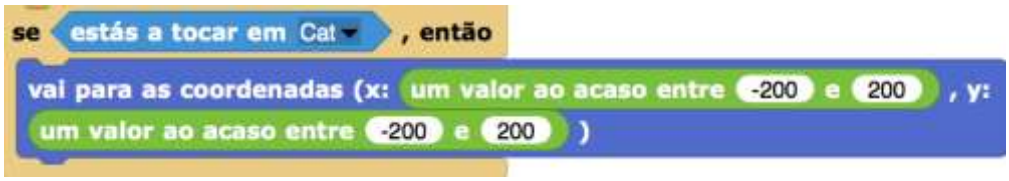
Os alunos programam o gato para se mover com as setas. Aqui têm de determinar o que acontece se o gato estiver no limite.

[Passo 3]

Os alunos têm que programar o rato para se movimentar aleatoriamente. Neste caso, a ideia é que o rato num loop ao acaso dê um número aleatório de passos e gire em graus aleatórios. Os alunos fazem isso com o bloco *Movimento/ anda [x]passos* e o bloco *Movimento/gira [x] graus* no qual eles inserem o operador um valor ao acaso*[x]to[y].*

[Passo 4]

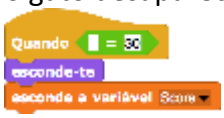
O próximo passo é programar o rato para se esconder quando toca no gato. A ideia é que o rato se esconda e apareça numa localização aleatória quando toca no gato. Neste caso, o jogo não acaba quando se apanha o primeiro rato. Os alunos podem adicionar a sua própria regra. Em qualquer caso, têm de usar o operador valor ao acaso*[x] e [y]*



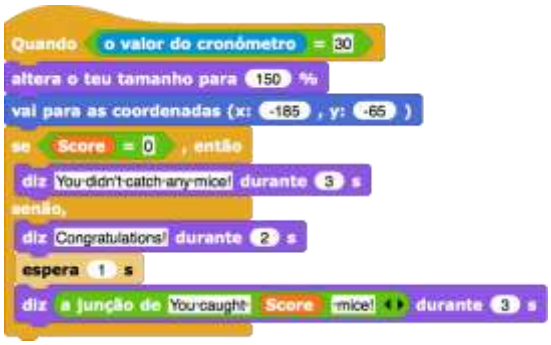
[Passo 5]

No caso de querermos saber o número de vezes que o rato foi apanhado, temos que adicionar o contador. Os estudantes criam uma nova variável- pontuar e adicionar ao código do gato. A pontuação no início do jogo tem sempre que ser zero. O estudantes fazem isso no bloco Variáveis/ altera para *[variável] para[x]*. Se quisermos que a pontuação seja mostrada ao jogador, os alunos têm de adicionar o bloco mostra variável. Depois, os alunos adicionam um novo controlo (*Controlo/Quando*) para verificar se o gato toca no rato. Se o gato tocar no rato, o resultado aumenta de 1 em 1. (*Variáveis/altera [pontuação]para [x]*).

[Passo 6]
Os alunos determinam quando é que o jogo acaba. Eles fazem isso ao adicionar o cronómetro. Depois de algum tempo (ex.: 30 segundos) o rato e o gato desaparecem, a variável pontuação é escondida e o jogo termina.



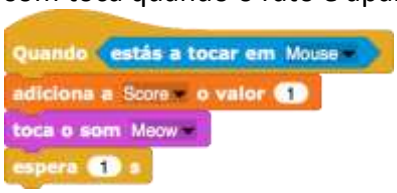
Os alunos têm de adicionar esses guiões aos blocos do gato e do rato.
[Passo 7]
Os alunos têm de programar a rapariga para resumir o nível de sucesso do jogador. Se o jogador não apanhar nenhum rato, a rapariga diz : " Não apanhaste nenhum rato!". Se apanhar, ela diz: "Parabéns! Apanhaste x ratos!"



[Tarefas adicionais]

Os alunos podem adicionar elementos ao seu jogo. Por exemplo, a rapariga que salta sempre que toca no rato.

Os alunos podem adicionar som. Por exemplo, adicionar o som do gato. O som toca quando o rato é apanhado.



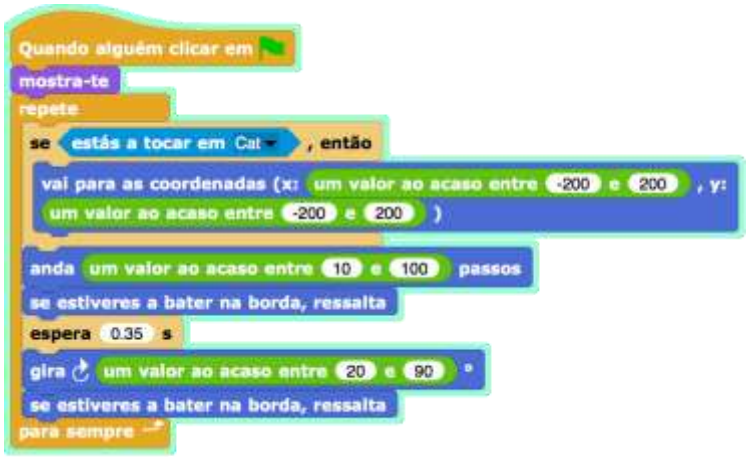
Reflexão e avaliação

Os alunos ajustam o código:

- o rato move-se entre 20 a 60 passos para sempre;
- o rato vai para a localização x = 100 quando toca no gato;
- o rato vira-se 90 graus sempre;
- etc;

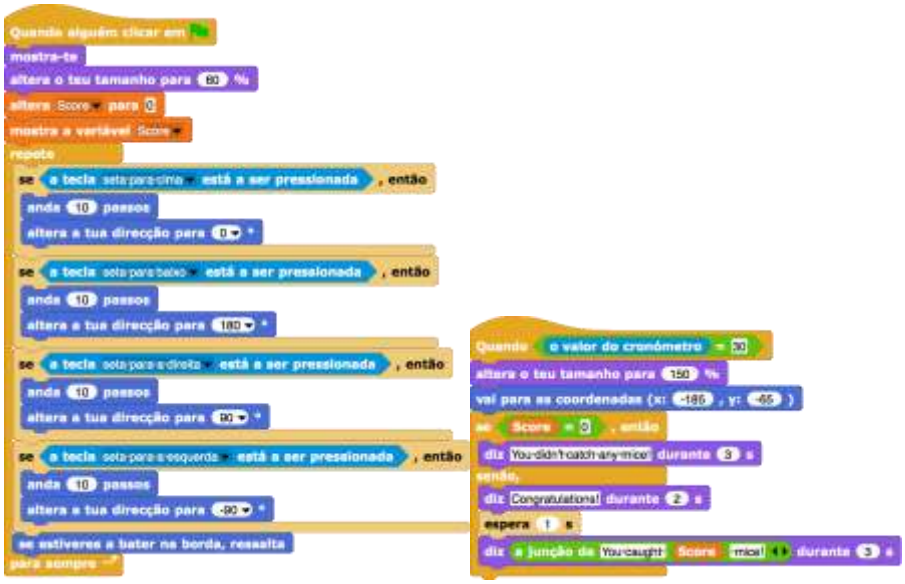
[Código Final]

O Rato



O Gato

A rapariga



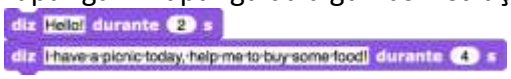
O Fundo

Instrumentos e Recursos para o Professor	• Toda a atividade no Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse • Website para imagens grátis: https://pixabay.com/ • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*.

	Ljubljana: MK.
Recursos/Materiais para os alunos	● Template in Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse_0 ● Website para imagens grátis: https://pixabay.com/ ● Instruções para os alunos (C4G15_InstructionsForStudent.docx)

Cenário de Aprendizagem 16 - Comprar comida para um piquenique

Cenário de Aprendizagem Título	Comprar comida para um piquenique
Experiência Prévia de programação	Adição de texto para a imagem (*sprite)* Mostrar e esconder os *sprites* Usar operadores Usar variáveis Usar concatenação de sequências Usar condições
Objetivos de Aprendizagem	Objetivos de Aprendizagem Gerais: ● Variáveis ● Condicionais ● Operadores Objetivos de aprendizagem específicos orientados para o pensamento algorítmico: ● Os alunos usam variáveis para definir o preço para as diferentes imagens (*sprites)* ● Os alunos mudam o valores das variáveis, já que o orçamento muda quando o jogador compra comida. ● Os alunos usam a condição if para verificarem o dinheiro disponível ● Os alunos usam operadores para junção de texto - valores das variáveis-texto ● Os alunos usam operadores para compararem preços e orçamentos ● Os alunos usam operadores (subtração) para alterar os valores das variáveis
Objetivos, Tarefas e Breve descrição das atividades	Breve descrição: A rapariga vai fazer um piquenique e precisa de ajuda para comprar alguma comida. Ela tem 15 euros e não pode gastar mais do que esse valor. Quando ela compra alguma coisa, o valor do orçamento muda. Se o orçamento for demasiado baixo, ela não poderá comprar a comida escolhida. Tarefa: Os alunos têm de programar 3 imagens (*sprites)* diferentes : a rapariga, a comida (que eles podem duplicar com diferenças ligeiras) e o botão de terminar. A rapariga dá instruções, diz quanto dinheiro o jogador tem e no fim (ao clicar no botão de terminar) ela diz quantos produtos saudáveis e quantos não saudáveis o jogador comprou. A comida diz o seu preço quando o rato passa por cima. Se o jogador tiver dinheiro suficiente, pode comprar um produto e o valor do orçamento muda. De outra forma, a comida não pode ser comprada. **Objetivo: Os alunos irão aprender a trabalhar com variáveis: definir diferentes valores de início, usar as condições para comparar os valores das variáveis, mudar o valor das variáveis, usar as variáveis para contar a comida saudável e não saudável. Além disso, eles irão repetir adição de texto, junção de texto e condições Se.**

Duração das atividades :	45 minutos
Estratégias e Métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem através de game-design e resolução de problemas.
Formas de ensino	Trabalho individual/ Trabalho em pares
Sumário da Lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) A rapariga está na loja a comprar comida para o piquenique. Ela tem 15 euros. Ela consegue ver o preço da comida quando o rato passa por cima e compra clicando na comida selecionada. as compras não devem superar o valor que ela tem disponível. Ao clicar no botão terminar, ela diz quantos produtos saudáveis e não saudáveis o jogador comprou.  [Passo 1] Esta atividade é para ser realizada individualmente ou em pares. O professor dá algumas sugestões, explica as partes mais difíceis e ajuda quando necessário. Os alunos escolhem o fundo e adicionam o *sprite* principal, ex. : a rapariga. A rapariga dá algumas instruções no início, ex.:  [Passo 2] Neste jogo, iremos precisar de algumas variáveis : ● *Orçamento, para definir a quantidade de dinheiro disponível* ● *healthy_food*, para contar quantos elementos saudáveis o jogador comprou ● *unhealthy_food*, para contar quantos elementos não saudáveis o jogador comprou ● uma variável para cada comida ex.: *watermelon_price*, para definir o preço de cada comida

No início, a variável orçamento é definida, por exemplo, nos 15 euros. As outras duas variáveis são definidas em 0. Este código pode ser adicionado antes do código da rapariga do Passo 1 .

[Passo 3]
Os alunos adicionam um *sprite* (comida) e escolhem os seus trajes.
O código Food's (watermelon's) necessita 3 eventos de controlo :

a) Quando a bandeira verde é clicada: para mostrar e definir o preço da comida.

Deixar que o preço da variável seja determinada de forma razoável (um valor superior a 1).

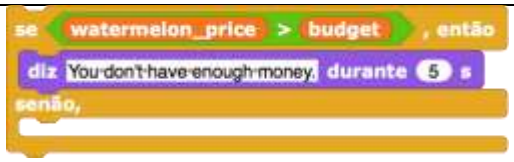
b) *Quando o rato entrar em ti:* para dizer ao jogador quanto é que os produtos custam

Os alunos podem usar *Looks* – o bloco pensa em junção ao texto - valor variável - texto ex..:

c) Quando clicam: aqui têm que fazer uma pequena reflexão

1) Em que caso o jogador pode comprar o produto e em que caso não pode?
2) O que acontece com o orçamento quando o jogador compra comida?
3) Como é que contamos os produtos comprados?
4) O que acontece com a comida na prateleira?

1) O jogador pode comprar o produto se tiver dinheiro suficiente. Então, os alunos têm que comparar duas variáveis: orçamento e watermelon_price.Se a melancia custar mais do que o orçamento do jogador, ele não a pode comprar. Os alunos podem adicionar algum texto para dizer ao jogador que não pode comprar aquele produto.



2) Se o jogador tem 15 euros e comprar a melancia por 4 euros, ele agora tem 15-4 =11 euros. Então, o valor do orçamento é agora :
valor do orçamento anterior – watermelon_price.

Os alunos podem adicionar alguma texto aqui também.

3) A contagem do número de produtos comprados vai ser feito com *healthy_food* variable by 1.

4) Quando a comida é selecionada, esconde-se

Uma solução possível é:

[Passo 4]
Para ter mais comida nas prateleiras, os alunos têm que duplicar o imagem da melancia. Se a segunda comida for um bolo. O código do [Passo 3] precisa de algumas mudanças.
Os alunos têm de :

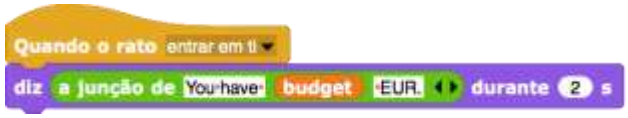
- Mudar o traje
- Criar uma nova variável : *cake_price*
- Definir o preço do bolo: *cake_price*
- Mudar o código de cada bloco de *watermelon_price* com *cake_price*
- Mudar a resposta ao comprar o bolo
- Mudar *change healthy_food by 1* to *change unhealthy_food by 1*

Ex.: . o código “quando o rato clicar em ti”para o bolo pode ser:

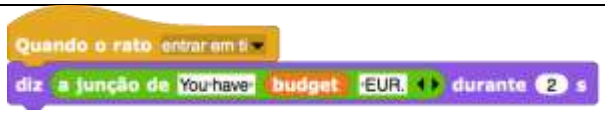
[Passo 5]
Quando o jogador acaba as suas compras, clica no botão Finish. Para dizer ao programa que o jogador clicou no botão (terminar de comprar comida), enviamos uma mensagem.

[Passo 6]
No fim, voltamos a imagem da rapariga.
Quando o jogador termina as suas compras, queremos que a rapariga lhe diga quantos produtos saudáveis e não saudáveis ele comprou.
Quando o jogador clica no botão terminar, uma mensagem é enviada.
Quando a rapariga recebe esta mensagem, ela diz, ex.: " Escolheste X produtos saudáveis e Y produtos não saudáveis"

[Passo 7]
A qualquer altura durante o jogo, o jogador pode verificar o seu orçamento ao clicar com o rato na rapariga. Ex.: ela pode dizer/pensar algo como:



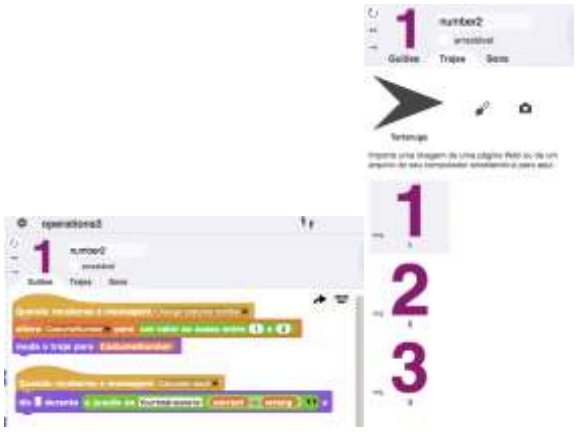
[Código Final]
Rapariga

	 Comida  Botão Terminar  [Tarefas Adicionais] Os alunos podem adicionar tarefas extra de acordo com os seus desejos ou podem seguir as tarefas abaixo: ● Mudar o jogo para se conseguir comprar cada comida 3 vezes. ● Dar mais dinheiro ao jogador no início. ● No fim, a rapariga diz também quantos produtos o jogador comprou Ex.: “Compraste 2x melancia, 1x uvas, 2x batatas fritas.”
Instrumentos e Recursos para o professor	● Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic ● Actividade em Snap! com tarefas adicionais (possíveis soluções): https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic%20%2B%20Add.%20Task ● Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček.* Ljubljana: Pasadena. ● Vorderman, C. (2017). *Računalniško programiranje za otroke.* Ljubljana: MK.
Recursos/Materiais para os estudantes	Instruções para o aluno (C4G16_InstructionsForStudent.docx)

Cenário de Aprendizagem 17 - Operações

Título do Cenário de Aprendizagem	Operações
Experiência de programação prévia	Usar variáveis para contar os pontos e escolher a imagem do stage e do *sprite* Usar números aleatórios para escolher o stage décor e a imagem do *sprite* Usar *loop* infinito Usar condições Usar operações para comparar Usar sensores para o diálogo(perguntar...e esperar) Usar transmissão de eventos
Objetivos de aprendizagem	Objetivos de aprendizagem gerais: ● Variáveis ● Condições ● *Loop* ● Sensores de blocos ● Transmissão de eventos Resultados de aprendizagem específicos orientados para o pensamento algorítmico: ● Os alunos usam variáveis para contar pontos e para adicionar os trajes do stage e dos *sprites* ● Os alunos usam variáveis para contar pontos ● Os alunos iniciam variáveis para a contagem de pontos ● Os alunos usam condicionais e operações lógicas ● Os alunos usam a transmissão de eventos para mudar o *sprite* e calcular o resultado final.
Objetivos, Tarefas e Breve descrição das atividades	Breve descrição: Vamos ver enquanto jogamos se o jogador dominou as operações aritméticas no Snap!. As regras são as seguintes: Durante 10 vezes gerar operações aritméticas onde o primeiro operante possua 6 números aleatórios e o segundo operante seja um número entre 1 e 3. O jogador tem que introduzir a resposta correta. As respostas certas e erradas são contabilizadas. No fim do jogo, o resultado correto é reportado. Tarefas: Os alunos têm que definir o cenário/ o *stage décor*/ e o traje do *sprite;* planear as variáveis requeridas, determinar os blocos que precisam. No fim, têm que criar os códigos para o *sprite* do stage. As tarefas adicionais podem ser: ● Configurar o *sprite*, dependendo do resultado, para dizer : “ Bom

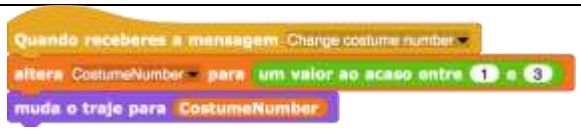
	para ti!" ou " Tu não sabes muito bem as operações aritméticas no *Snap! ainda!"* **Objetivo: Os alunos irão melhorar os seus conhecimentos previamente adquiridos sobre variáveis, números aleatórios, *loops*, e transmissões.**
Duração das Atividades	45 minutos
Estratégias e Métodos de Ensino e aprendizagem	Aprendizagem ativa(discussões, experiências com um jogo preparado previamente),aprendizagem através de game-design, resolução de problemas
Formas de ensino	Trabalho individual/ Trabalho em pares Trabalho presencial com a turma toda
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e Avaliação) O professor coloca o programa correspondente à necessidade de um jogo que determine se as operações aritméticas no Snap! foram bem dominadas e demonstra o projeto. https://snap.berkeley.edu/project?usr=ddureva&project=operations3.  1. O professor discute como formular a condição da tarefa. A tarefa é formulada. 2. As variáveis são comentadas, assim como, a forma como são definidas, iniciadas e mudadas. As operações aritméticas serão geradas 10 vezes, e o primeiro operante possui 6 números aleatórios e o segundo operante seja um número entre 1 e 3. O jogador tem que introduzir a resposta correta. As respostas certas e erradas são contadas.O resultado é reportado no fim do jogo.

3. Comandos de números aleatórios, operações lógicas e aritméticas e revisão dos comandos de transmissão de eventos.
4. É debatido se o código é para o stage ou para o *sprite*. No exemplo, o código principal é para o cenário, e o código do *sprite* tem guiões para mudar o traje e calcular o resultado final. ,
5. 

Código para o cenário	Cenário/Stage Décor/
	

O código do cenário contém as inicializações das variáveis para as respostas corretas e erradas.

Para selecionar uma operação os seguintes comandos são utilizados:



A escolha do traje para o *sprite* é feito através da transmissão do Número *Sprite*. O traje selecionado é guardado na variável do número do traje, que é definido para todos os objetos no projeto e, por isso, é usado no código do stage.

Uma vez que o cenário/ *stage décor*/ e o traje do *sprite* tenham sido selecionados aleatoriamente, questão é direcionada ao jogador e para introduzir a resposta correta para a operação, utilizamos o seguinte comando:

A resposta introduzida é comparada com o resultado das operações selecionadas.

O seguinte comando é usando:

condição if

senão 6

Se a operação "-" é selecionada, depois é verificado se o resultado é 6 - *"O número da variável do traje do sprite"* corresponde à resposta. Se corresponder, a variável correta aumenta, de outra forma, a variável da contagem de respostas erradas aumenta.

Para o resto dos comandos o guião é similar, a diferença é na operação selecionada.

Para evitar repetir a ordem do código para o resto das operações, os alunos podem ter que ser ensinados como copiar parte do código e mudar as operações aritméticas em : 

Copiar código:

1. Clicar com o botão direito do rato no guião
2. Escolher duplicado

	3. Usar o rato para colocar o duplicado no guião na localização correspondente. Ao critério dos professores, os alunos podem ter como tarefas perceber como copiar o código de forma autónoma. Mudar a operação. 1. Clicar com o botão direito do rato na placa da operação. O contexto do menu vai aparecer.  2. Escolher Mudar para outro bloco. Uma lista de operações irá aparecer.  3. Escolher a operação Nota: Se a idade e conhecimento dos alunos de operações aritméticas permitir, as tarefas podem ser alargadas com operações de exponenciação e operações de módulos(mod). 4. Os alunos trabalham em equipas para criar o seu próprio cenário/ stage décor/ e trajes para o *sprite*. Se houver constrangimentos de tempo, podem usar um projeto pré-construído que contenha o stage e o *sprite*.
Instrumentos e Recursos para os professores	Toda a atividade in Snap!: https://snap.berkeley.edu/project?user=ddureva&project=operations3 Toda a atividade in Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)
Recursos/ Materiais para os estudantes	Atividade Pré-construída in Snap! https://snap.berkeley.edu/project?user=ddureva&project=operations_half ● Instruções para os estudantes (C4G17_InstructionsForStudent.docx)

Cenário de Aprendizagem 18 - Reciclar

Título do Cenário de Aprendizagem	Reciclar
Experiência prévia de aprendizagem	Mostrar e esconder um *sprite* Usar variáveis para contar pontos Usar *loop* infinito Usar condicionais Usar operações para comparar Usar sensores de cores
Objetivos de aprendizagem	Objetivos de aprendizagem gerais: ● Variáveis ● Condicionais ● *Loop* ● Apontar em direção ● Sensores de blocos ● Refatoração de códigos Resultados de aprendizagem específicos orientados para o pensamento algorítmico: ● Os alunos usam *esperar até que and operações lógicas* até ao fim do jogo. ● Os alunos usam *esperar até , e blocos* para mudar o stage ● Os alunos usam variáveis para contar os pontos ● Os alunos usam condicionais e operações lógicas ● Os alunos comparam códigos de *sprites* similares ● Os alunos usam refatoração de códigos ● Os alunos usam a posicionamento dos *sprites* (numa tarefa adicional usam posicionamento aleatório).
Objetivos, Tarefas e Breve descrição das atividades	Breve descrição: Alguém deixou lixo em frente à escola. É pedido ao jogador para ajudar a organizar a recolha do lixo, ao separar na reciclagem o papel e o vidro. Quando o lixo é colocado no contentor correto, o lixo é escondido. Se o lixo for colocado no contentor errado, a mensagem - " Este não é o contentor de papel" ou " Este não é o contentor de vidro" aparece e o lixo regressa à sua posição original. O jogo acaba quando todo o lixo é colocado nos contentores corretos. Tarefa: Os alunos têm que explorar os códigos do *stage* e *sprites*, comparar códigos de desperdício-papel e desperdício-vidro dos *sprites*, adicionar novos *sprites* e guiões, e alterar o guião no palco respeitando os novos *sprites* adicionados.

	As tarefas adicionais podem ser: ● Mudar a posição do *sprite* do desperdício com uma escolha aleatória das coordenadas dos *sprites*. ● Diminuir o número de palcos e extrair o robô como um *sprite* separado. (O robô faz parte do fundo do stage). **Objetivo: Os alunos irão melhorar o conhecimento previamente adquirido e irão ampliar o cenário do jogo com novos objetos, código e mudança de códigos respeitando os novos *sprites.* Eles serão treinados em refatoração de código.**
Duração das Atividades	45 minutos
Estratégia e métodos de ensino e aprendizagem	Aprendizagem ativa (discussão, experiências com um jogo previamente preparado), aprendizagem baseada em game-design, resolução de problemas.
Formas de ensino	Trabalho individual/ Trabalho em pares/ Trabalho individual com toda a turma
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) 1. O professor expõe o problema da separação da recolha de lixo e comenta as cores dos contentores para os diferentes tipos de lixo - azul para o papel, verde para o plástico. 2. Diz aos alunos para jogar e descrever em palavras: Quantos cenários eles assistem e quantos *sprites*(personagens) ? Como o jogo começa? Qual o *sprite que* pede o nome do jogador? Quantas variáveis são usadas e como se chamam? O que acontece quando o papel é colocado no contentor do vidro e quando é colocado no contentor do papel?  1. Atualização dos comandos estudados

Os comandos para entrar num diálogo com o utilizador são relembrados. É feito um comentário sobre a mudança de cenários- Cenário 1 com o Robô, Cenário 2 com a escolha e o lixo e Cenário 3 com o Robô e a descrição Bravo! São discutidos possíveis comandos para alterar cenários.

É discutido que a verificação para a colocação adequada do lixo no contentor deve ser feita com um bloco condicional e blocos com condições do grupo de sensores. É dada uma descrição oral : Se um lixo de papel tocar no contentor do papel, o lixo é escondido (colocado no contentor correto) e os pontos relacionados a recolha do lixo são aumentados de 1 em 1. Se o lixo de papel tocar no contentor do vidro, diz - *"Este não é o contentor do papel".* O mesmo acontece para o lixo de vidro.

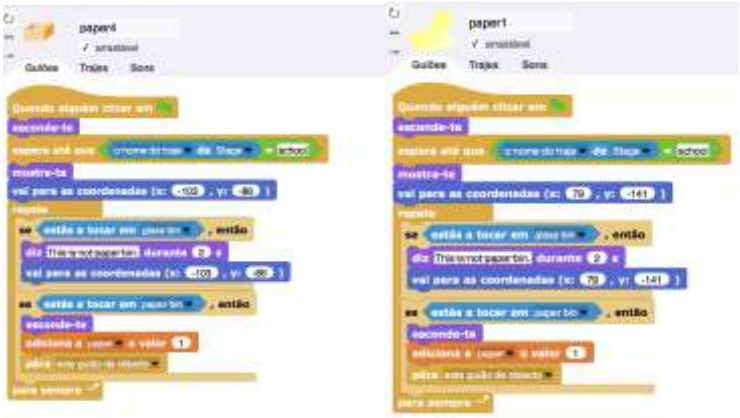
2. Examinar o código do cenário e das personagens

Depois de discutir as possibilidades para a resolução do problema, os códigos para os cenários e para as personagens são discutidos.

Os códigos dos cenários são comentados com ênfase em:

- Definir o valor inicial do nome da variável e usá-lo num diálogo com o utilizador;
- Mudar o cenário do stage (trajes) e a condição para o fim do jogo;

Quando estamos a olhar para o código de uma personagem, é aconselhável mostrar-lhes num único slide ou dar impresso o código dos pedaços de lixo de papel e do lixo de vidro. A comparação é feita entre códigos comuns e diferentes elementos nos códigos.



3. Definir uma tarefa para completar o jogo com dois novos *sprites* - lixo de papel e lixo de vidro, atribuindo um código a cada um e mudando o código do stage e do contentor de lixo.

	É discutido como criar os dois novos *sprites*. Opções- Duplicar os existentes e editar no *Snap!*, criar novos num editor de gráficos, ou procurar imagens gratuitas na Internet e importá-las para o jogo. Também é necessário comentar as mudanças nos códigos do stage respetivos à finalização do jogo. É também necessário discutir se é possível definir os valores iniciais das variáveis não nos códigos dos dois contentores, mas no código dos cenários e fazer os ajustes adequados. Ao critério dos professores, a condição da tarefa pode ser complicada: ● O lixo deve ser distribuído em qualquer sítio apropriado quando o jogo começar. Nota se que aqui as coordenadas nas quais o lixo pode ser disperso deve limitar para que esteja num sítio realista. Por exemplo, delimitado pelas coordenadas do re  tângulo vermelho. ● Introduzir um novo *Sprite do Robô* e reduzir o número de elementos do stage no palco. ● Escrever o código apropriado do Robô para que possa entrar em diálogo com o jogador em vez do sprite do contentor blue.
Tools and Resources for the Teacher	Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling Toda a atividade Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)

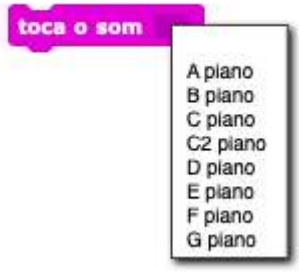
Resources/materials for the Students	● Atividade pré-construída em Snap! https://snap.berkeley.edu/project?user=ddureva&project=recycling ● Instruções para os estudantes (C4G18_InstructionsForStudent.docx)

Cenário de Aprendizagem 19.1 - Tocar piano

Título do Cenário de Aprendizagem	Tocar piano
Experiência prévia de aprendizagem	Usar variáveis para contar pontos; Usar o evento "Quando é pressionada!"; Usar *loop* em repetição; Usar condicionais; Usar transmissão de eventos para mudar de cenário/ stage décor/ e gerir as atividades do *sprite*;
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Variáveis ; ● Condicionais; ● *Loop*; ● Transmissão de eventos; ● Sons; ● Programar música; Resultados de aprendizagem específicos orientados para o pensamento algorítmico: ● Os alunos usam variáveis para a contagem de pontos; ● Os alunos iniciam variáveis para a contagem de pontos; ● Os alunos usam condicionais para estimar os pontos alcançados; ● Os alunos usam a transmissão de eventos para mudar de cenário/ stage decór/ e para atividades dos *sprites;* ● Os alunos usam blocos do grupo do som para compôr melodias; ● Os alunos identificam a necessidade de repetição do *loop* para diminuir o número de blocos nos guiões; ● Os alunos alargam a funcionalidade do jogo;
Objetivos, Tarefas e Breve Descrição das Atividades	Breve descrição: Vamos entrar no mundo maravilhoso da Rainha Maria. Ela convida o jogador para ouvir música no seu palácio. No salão de baile, o seu pequeno amigo dinossauro Dino toca piano. No jogo, Dino toca algumas melodias e os jogadores têm que adivinhar qual é a melodia. Se acertarem, ganham um ponto pela resposta correta, se não souberem a resposta, é-lhes retirado um ponto pela resposta errada. Depois de identificarem as notas musicais, é definida uma tarefa mais complexa: Dino toca uma melodia, e o jogador tem que identificar qual é a música . Por cada melodia identificada, o jogador recebe 5 pontos.

	Tarefa: Os alunos usam um cenário/ stage décor/ e trajes dos *sprites* pré-construídos. Eles precisam de planear as variáveis necessárias, determinar que blocos necessitam, conhecer os blocos do grupo do Som e a forma de tocar as notas. Criar guiões para tocar diferentes melodias. **Objetivo: Os alunos irão aprender sobre codificação de melodias e tocar e irão melhorar o conhecimento previamente adquirido sobre variáveis, *loops*, condicionais, transmissões e o outros eventos.**
Duração das Atividades	90 minutos
Estratégias e métodos de Ensino e Aprendizagem	Aprendizagem ativa(discussões, experiências com jogos previamente preparados), aprendizagem baseada em game-design, resolução de problemas.
Formas de Ensino	Trabalho individual/ Trabalho em pares/ Trabalho individual com toda a turma
Sumário da Lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) 1. O professor define a tarefa de criar o jogo. Os meios pela qual cada tarefa pode ser completada são discutidos. É concluído que os alunos não têm atualmente conhecimento dos recursos de programação disponíveis para programar uma melodia. 2. O professor demonstra parte do jogo, programando uma melodia. https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1  3. O professor mostra o código e explica como é que o comando do grupo do Som pode ser usado. No Snap! os sons da biblioteca embutida podem ser usados, assim como ficheiros do computador e as melodias de músicas tocadas em vários instrumentos.

Para selecionar um instrumento, usar o comando:



Nota: Existem muitos instrumentos no Scratch.

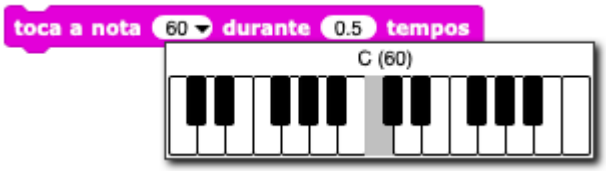
Os alunos testam o som de vários instrumentos.

4. O professor explica como definir as notas musicais.

O comando 

é usado. Nele, o primeiro número define a nota, e o segundo número descreve quanto tempo a nota vai ser tocada.

Quando se clica na seta a seguir ao primeiro número, aparece um teclado de um piano e a nota pode ser selecionada através dele. Este teclado de piano abrange duas oitavas.

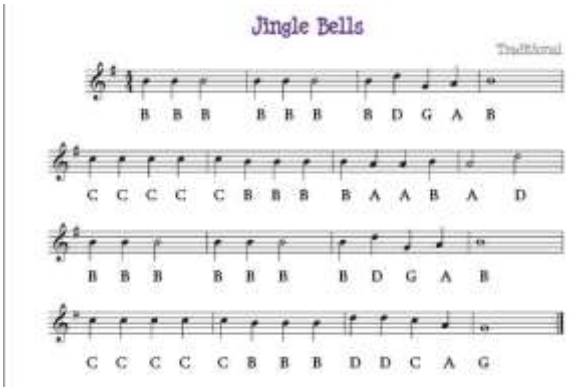


C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B	C
60	61	62	63	64	65	66	67	68	69	70	71	72

A duração de cada nota é definida pelos números 1 - nota completa- 0.5- metade, 0.25 - um quarto. (Para os alunos que não tenham estudado número reais, as frações decimais podem ser apresentadas na forma de frações ordinárias: ½, ¼, 1/8, etc.) .

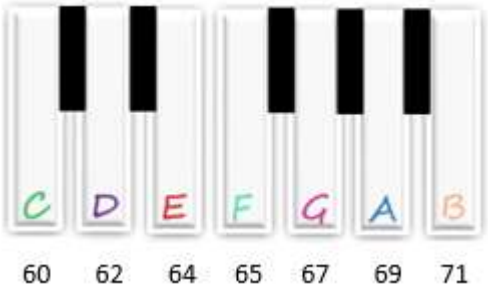
Ao critério do professor, os alunos podem experimentar comandos e estabelecer as dependências eles mesmos.

5. O guião da melodia do *Jingle Bells* é discutida ao usar a notificação

	musical.  6. A tarefa é definida para reduzir o número de linhas no código que são repetidas. É discutido o comando a ser usado (repetir *loop*) Os alunos são divididos em equipas que têm que criar o jogo, definido no início da lição. Cada equipa discute o cenário do jogo e descreve o plano do jogo na folha de descrição (*Attached SNAP_Program_Design_and_Planning Worksheet.docx*). As tabelas podem ser adicionadas à descrição para uma descrição detalhada de ações nos *stages* e *sprites*. A condição de o dinossauro dançar enquanto toca pode ser adicionada.(O dinossauro tem vários trajes nos ficheiros pré-preparados). 7. O professor pode mostrar várias partes de cenários do ficheiro. https://snap.berkeley.edu/project?user=ddureva&project=PlayAPiano
Instrumentos e Recursos para o professor	Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1 https://snap.berkeley.edu/project?user=ddureva&project=PlayAPiano
Recursos/materiais para os alunos	● Atividade pré-construída em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_Half_backed ● Instruções para os alunos (C4G19.1_InstructionsForStudent.docx)

Cenário de Aprendizagem 19.2 - Tocar piano

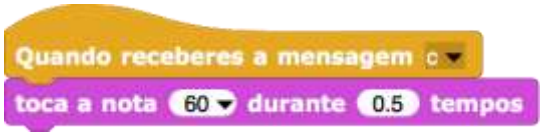
Título do Cenário de aprendizagem	Tocar piano
Experiência prévia de programação	Usar *loop* em repetição Usar variáveis Usar condicionais
Objetivos de aprendizagem	Objetivos de aprendizagem: • Condicionais • *Loops* Objetivos de aprendizagem específicos orientados para o pensamento algorítmico: • Os alunos usam o repetir *loop* para tocar música • Os alunos usam código para fazer com que os *sprite* reajam aos inputs • Os alunos adicionam som ao *sprite* • Os alunos usam código para mudar o traje do *sprite*.
Objetivo, tarefa e breve descrição das atividades	Breve descrição : Os alunos têm de tocar uma música no piano de acordo com as notas dadas. Tarefas: Os alunos devem programar as teclas do piano - cada tecla tem que tocar uma melodia em concreto. No *stage*, dois botões diferentes têm de ser mostrados, um para mostrar as notas e outro para tocar a melodia. **Objetivo: Os alunos irão aprender como tocar música e mudar o traje ao clicar num *sprite*.**
Duração das Atividades	45 minutos
Estratégia e Métodos de ensino e aprendizagem	Aprendizagem ativa, aprendizagem baseada em game-design, resolução de problemas.
Formas de ensino	Trabalho individual/ Trabalho em pares

Sumário da Lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) No início, é mostrado um piano no palco. Ao lado do piano deve ter dois botões. Ao clicar no primeiro botão deve aparecer as notas e as palavras da música, ao clicar no segundo botão deve tocar a melodia que precisa ser repetida. Adicionalmente, ao lado do piano deve haver um botão "X", que irá reiniciar o projeto. [Passo 1] Abrir o programa *Abrir um piano*. O programa contém todos os fundos e *sprites* necessários para esta tarefa. O fundo é dado, e também o *sprite* para a tecla C e uma tecla preta. Os alunos têm de duplicar a tecla C, movê-la para a posição correta e alterar o nome. As teclas devem estar na seguinte ordem: C, D, E, F, G, A, B. O teclado devem estar de acordo com a figura seguinte e reproduzir a melodia escrita abaixo das teclas:  Duplicar o *sprite* "black_key" 4 vezes para ter 5 teclas pretas, e nomeá-las de tecla 2 a tecla 5. Colocar novas teclas pretas entre teclas D e E, F and G, G and A, and A and B. Se a tecla preta estiver escondida atrás das teclas brancas, usar o código seguinte:  Fazer o mesmo para a tecla B e colocá-las no fim do teclado. Desmarcar o botão "arrastável", para que os *sprites* das teclas não possam ser movidos enquanto está a tocar. 

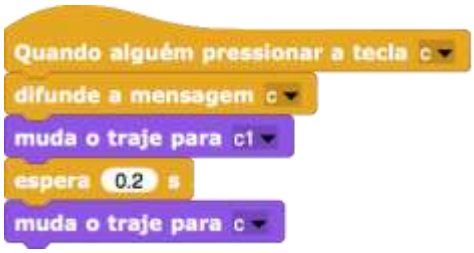
[Passo 2]

Ativar a reprodução de sons ao pressionar *sprites*. Para a tecla "C", adicionar bloco de início "Quando alguém pressionar a tecla" e permitir a transmissão da mensagem "c"

Para produzir som quando uma tecla é pressionada, adicionar o bloco de início "Quando receberes a mensagem c", e adicionar uma nota de reprodução 60 para 0.5 batidas.



Para destacar qual a tecla pressionada, o traje do *sprite* deve ser alterada temporariamente. Importar traje c1 ao *sprite* C. No bloco" Quando alguém pressionar a tecla", alterar o traje c1 por 0.2 segundos, depois retomar ao traje c.



[Passo 3]

Repetir o passo dois para as restantes teclas brancas.

[Passo 4]

Para tocar o piano usando o teclado, adicionar o bloco "Quando alguém pressiona a tecla" ao *sprite* da tecla c, e copiar o resto do código do bloco "Quando o rato clicar em ti"

Se a tecla C no teclado estiver para baixo, o som irá repetir-se enquanto

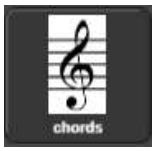
a tecla é pressionada. Isto acontece porque a mensagem “a” está em transmissão repetidamente. Para parar de transmitir uma mensagem, no fim do código adicionar o bloco “espera até que” do separador Controlo.

Para terminar de transmitir uma mensagem, usar o operador "é falso que " e adicionar o bloco “ a tecla está a ser pressionada”.

Fazer o mesmo para as restantes teclas brancas.

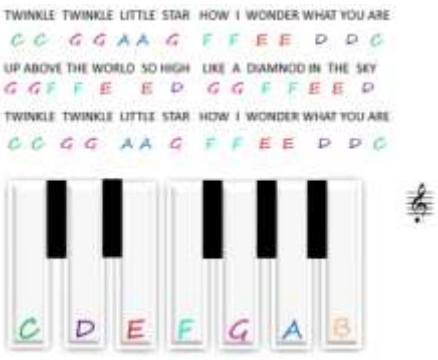
[Passo 5]

Criar um novo *sprite* e importar uma imagem de uma tecla de violino como traje. Este será um botão para exibir as palavras e as notas a serem tocadas.



Para exibir as notas, permitir a transmissão da mensagem dos chords quando se clica no botão.

Importar um novo traje dos chords para o stage.



Adicionar o código que permita o *stage* mudar os trajes para chords

	quando recebe a mensagem chords.  [Passo 6] Encontrar o *sprite* com as notas como traje. Este botão irá tocar a música que precisa ser repetida.  O código é escrito para os primeiros dois versos, e tu tens que escrever o código para os outros versos. É a mesma música que é exibida na pauta.


Quando o rato clicar em ti
repete 2 vezes
toca a nota 60 durante 0.5 tempos
repete 2 vezes
toca a nota 67 durante 0.5 tempos
repete 2 vezes
toca a nota 69 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 65 durante 0.5 tempos
repete 2 vezes
toca a nota 64 durante 0.5 tempos
repete 2 vezes
toca a nota 62 durante 0.5 tempos
toca a nota 60 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 67 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
toca a nota 65 durante 0.5 tempos
toca a nota 65 durante 0.5 tempos
toca a nota 64 durante 0.5 tempos
toca a nota 64 durante 0.5 tempos
toca a nota 62 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 60 durante 0.5 tempos
repete 2 vezes
toca a nota 67 durante 0.5 tempos
repete 2 vezes
toca a nota 69 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 65 durante 0.5 tempos
repete 2 vezes
toca a nota 64 durante 0.5 tempos
repete 2 vezes
toca a nota 62 durante 0.5 tempos
toca a nota 60 durante 0.5 tempos

[Passo 7]

Criar um novo botão X que irá reiniciar o projeto (sem as notas musicais).

Criar um novo *sprite* - reiniciar, escolher o traje “X” e definir o tamanho até 50%. Permitir a transmissão da mensagem "em branco " quando o botão é pressionado.

Adicionar o bloco de início “Quando receberes a mensagem” ao stage para mudar o traje para "em branco" depois de receber a mensagem "em branco"

[Tarefas adicionais]

Os alunos podem adicionar tarefas adicionais de acordo com os seus desejos ou seguir as tarefas abaixo:

- Duplicar a imagem (*sprite)* nota musical (e mudar a sua posição no fundo) e escrever um programa para outra música.
- Adicionar um cenário com acordes para uma nova música.

[Código final]

Uma tecla

	A tecla do violino  Nota Musical


Quando o rato clicar em ti
repete 2 vezes
toca a nota 60 durante 0.5 tempos
repete 2 vezes
toca a nota 67 durante 0.5 tempos
repete 2 vezes
toca a nota 69 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 65 durante 0.5 tempos
repete 2 vezes
toca a nota 64 durante 0.5 tempos
repete 2 vezes
toca a nota 62 durante 0.5 tempos
toca a nota 60 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 67 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
toca a nota 65 durante 0.5 tempos
toca a nota 65 durante 0.5 tempos
toca a nota 64 durante 0.5 tempos
toca a nota 64 durante 0.5 tempos
toca a nota 62 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 60 durante 0.5 tempos
repete 2 vezes
toca a nota 67 durante 0.5 tempos
repete 2 vezes
toca a nota 69 durante 0.5 tempos
toca a nota 67 durante 0.5 tempos
espera 0.1 s
repete 2 vezes
toca a nota 65 durante 0.5 tempos
repete 2 vezes
toca a nota 64 durante 0.5 tempos
repete 2 vezes
toca a nota 62 durante 0.5 tempos
toca a nota 60 durante 0.5 tempos

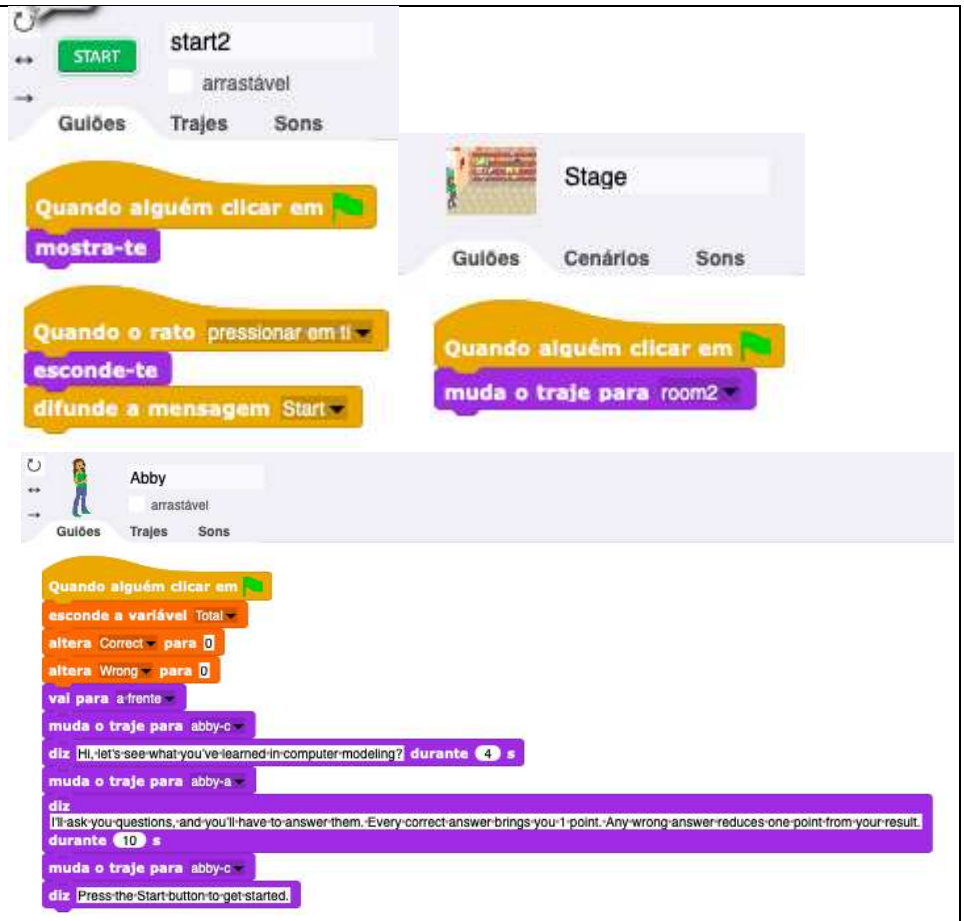
	X   O stage  
Instrumentos e recursos para o Professor	Snap! projecto “Tocar piano”: https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20a%20Piano

Recursos/Materiais para os alunos	Atividade pré-construída em Snap!:
	https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20Piano (27.1.2020)
	Imagens:
	• Imagens Sprites:
	  • a.png, a1.png
	  • b.png, b1.png
	  • violin_key.png
	  • Cenários: notes.png

Cenário de Aprendizagem 20 - Teste

Título do Cenário de Aprendizagem	Teste
Experiência Prévia de Programação	Mostrar e esconder o *sprite (*imagem) Usar variáveis para a contagem de pontos Usar o *loop* infinito Usar condicionais Usar operações para comparação Usar sensores de cores Mudar o stage
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● Variáveis ● Condicionais ● *Loop* ● Bloco de sensores Objetivos de aprendizagem específicos orientados para o pensamento algorítmico: ● Os alunos usam condicionais para estimar a resposta - Correta ou Errada ● Os alunos usam blocos para mudar o traje do *stage* ● Os alunos usam variáveis para a contagem de pontos ● Os alunos usam operações lógicas ● Os alunos usam editores de gráficos externos para preparar cenários complexos dos *stages*
Objetivo, Tarefas e Breve Descrição de Tarefas	Breve descrição: Ajuda o teu professor a testar o teu conhecimento no Snap! através da criação de um jogo baseado em missões para testar os comandos usados no Snap Tarefa: Os alunos têm que explorar o jogo de exemplo, escolher um jogo pré-construído, encontrar ou criar o seu próprio *sprite* que irá definir as perguntas, escolher a partir do jogo pré-construído ou criar o cenário do *stage* inicial e cenário dos *stages* com questões apropriadas, modificar e alargar guiões em testes respeitando as questões. **Objetivo: Os estudantes irão melhorar o conhecimento previamente adquirido e irão alargar o cenário do jogo com um novo cenário, código e mudança de código com o respectivo *stage.***
Duração das Atividades	90 minutos
Métodos e	Aprendizagem ativa (discussões, experiência com um jogo previamente

Estratégias de Ensino e Aprendizagem	preparado), aprendizagem baseada em game-design, resolução de problemas.
Formas de Ensino	Trabalho individual/ Trabalho em pares Trabalho presencial com toda a turma
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e Avaliação) 1. O professor levanta o problema da necessidade de criar um jogo-teste para testar o conhecimento de programação. 2. Coloca os alunos a jogar e descrever em palavras: Quantas decorações do palco eles observam ou quantos *sprites* (imagens)? Como o jogo começa? Quantas variáveis são usadas, como são designadas, para quê que são usadas? O que acontece quando a resposta é correta/errada? Quantas questões são apresentadas no teste? /trabalho individual ou trabalho em pares ao critério do professor 3. Comentar o algoritmo de perguntar e responder às questões /atividade presencial • Mover para um traje do stage (contém a questão); • Atribuir à Abby um traje para perguntar uma questão; • Abby diz - Responde Sim ou Não; • O jogador introduz uma resposta - Sim ou Não; • Se a resposta for correta, a Abby diz “Correto” e o número de respostas corretas aumenta; De outra forma, a Abby diz “ Estás errado” e o número de respostas incorretas aumenta. 4. Comenta o que acontece depois de responder a todas as questões /atividade presencial/ • Mudar o traje/cenário do stage; • Abbey indica o número de respostas corretas e erradas e dá uma estimativa; 5. Examinar os códigos no jogo / Atualizar o conhecimento antigo/ Atividade individual e presencial Os comandos para começar o diálogo com o utilizador, para mudar o stage décor e o traje da personagem, os comandos condicionais são comentados. Os códigos para cada personagem são examinados. A criação da variável é comentada.


START
start2
arrastável
Guiões
Trajes
Sons
Quando alguém clicar em
mostra-te
Stage
Guiões
Cenários
Sons
Quando o rato pressionar em ti
esconde-te
difunde a mensagem Start
Quando alguém clicar em
muda o traje para room2
Abby
arrastável
Guiões
Trajes
Sons
Quando alguém clicar em
esconde a variável Total
altera Correct para 0
altera Wrong para 0
vai para a frente
muda o traje para abby-c
diz Hi, let's see what you've learned in computer modeling? durante 4 s
muda o traje para abby-a
diz I'll ask you questions, and you'll have to answer them. Every correct answer brings you 1 point. Any wrong answer reduces one point from your result. durante 10 s
muda o traje para abby-c
diz Press the Start button to get started.

Situações onde a resposta correta é sim e situações onde a resposta correta é não são comentadas.

O código para a classificação é discutido em detalhe assim como o porquê da variável Total é usada.

A forma como se desenha o cenário do stage para questões individuais é discutida.

Porque no Snap! não é possível escrever texto nos trajes e cenários, é necessário usar um editor gráfico externo. Outra opção é usar o MS Powerpoint para criar a questão e exportar a caixa de texto no formato de gráfico.

Inserir um traje no Snap! Pode ser revisto.

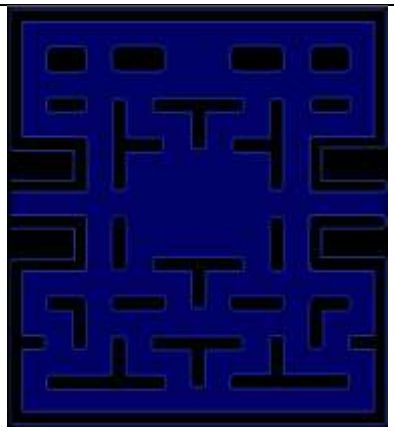
1. Dividir o grupo em equipas de 2 ou 3 alunos.

	2. Colocar o tópico para as questões do teste. Por exemplo, - Usar Variáveis; *Loops*; Movimentar-se, Sensores, Operações lógicas e Aritméticas 3. Criar as cenas com questões sobre um tópico desenvolvido pela equipa respetiva. Se necessário, o professor aconselha os alunos sobre o conteúdo das questões. As questões são discutidas e cada equipa criar um cenário para pelo menos duas questões. 4. Criar o código. Um ficheiro pré-preenchido de trajes de stages (cenários) e sprites (imagens) é dado aos estudantes para usarem. Eles podem também criar o seu próprio ficheiro. O trabalho é feito por analogia com um teste modelo.
Instrumentos e Recursos para o Professor	Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=test2 Toda a atividade em Scratch: ● Дурева Д., М. Касева, Г. Тупаров, Компютърно моделиране, 4. клас, Просвета, 2018, София (Dureva, D., M. Kaseva, G. Tuparov, Kompyutarno modelirane, 4. klas, Prosveta, 2019, Sofia)
Recurso/materiais para os alunos	● Atividade pré-construída em Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_en_tmp ● Instruções para os alunos (C4G20_InstructionsForStudent.docx)

Cenário de aprendizagem 21- Jogo simplificado do PACMAN

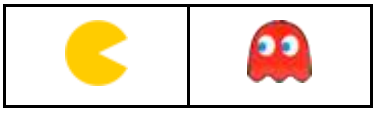
Título de cenário de aprendizagem	Jogo simplificado do PACMAN
Experiência prévia de programação	• Condicionais, • Programar múltiplos objetos, • Sensores singulares • *Loops* (infinito, repetir até), • Eventos baseados no movimento de objetos, • Números aleatórios
Objetivos de Aprendizagem	Objetivos de aprendizagem gerais: • Clonar um objeto, • Definir o comportamento do clone, • Transmissão de mensagens, • Leitura de valores booleanos nas expressões lógicas, • Definir, diferenciar, verificar de forma dinâmica e responder a dois diferentes estados do jogo, Objetivos de aprendizagem específicos orientados para o pensamento algorítmico: • Os alunos implementam os movimentos dos objetos com as setas, usando eventos e tendo em conta as restrições, • Os alunos usam clones para student uses clones para fazer exemplos do objeto original, • Os alunos sabem como programar o comportamento de cada clone, • Os alunos sabem o significado de mandar mensagens, • Os alunos implementam o envio da mensagem do clone para incrementar o contador, • Os alunos sabem como detectar que a mensagem foi recebida pelo objetivo e criar uma resposta apropriada.
Objetivo, Tarefas e Breve Descrição das Atividades	Breve descrição: Programar o jogo no qual a personagem principal vai recolher estrelas posicionadas aleatórias e ser perseguida por um fantasma. Tarefas: Os alunos têm de programar o movimento da personagem principal para que se mova dentro de um labirinto. Eles têm que

	implementar restrições no movimento para que a personagem principal não se possa mover através das paredes. De seguida, têm que programar o objeto estrela que irá clonar-se quando o jogo começa e depois numa nova localização aleatória cada vez que a personagem a coletar. Eles têm que definir o valor das estrelas recolhidas e terminar o jogo quando o jogador recolher 20 estrelas. Para tornar o jogo mais interessante, eles podem programar um fantasma malvado que se irá mover aleatoriamente ao longo do labirinto. Se o jogador tocar no fantasma, o jogo acaba. **Com esta atividade os estudantes irão rever o seu conhecimento sobre movimento dentro de um labirinto com o uso de blocos de sensores de cores que aprenderam nas atividades anteriores. Eles serão introduzidos ao conceito de clonar objetos com posições restritas e como criar uma personagem não-jogadora simples com o seu próprio movimento aleatório.**
Duração das atividades	90 minutos
Estratégias e métodos de ensino e aprendizagem	aprendizagem ativa, aprendizagem colaborativa, resolução de problemas
Formas de ensino	Aprendizagem Presencial Trabalho individual/ Trabalho em pares/ Trabalho de grupo
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) O jogador está a recolher estrelas posicionadas de forma aleatória enquanto é perseguido por um fantasma vermelho. Se o jogador e o fantasma colidirem, o jogo acaba. Se o jogador recolher 20 estrelas ele ganha. [Passo 1] Instruímos os alunos a desenharem um labirinto cuja área que é permitido que o jogador se mova é de uma cor (por exemplo: azul) e as paredes que param o movimento do jogador são de outra cor (por exemplo: preto). Para poupar tempo, podemos preparar a imagem do fundo do labirinto previamente.



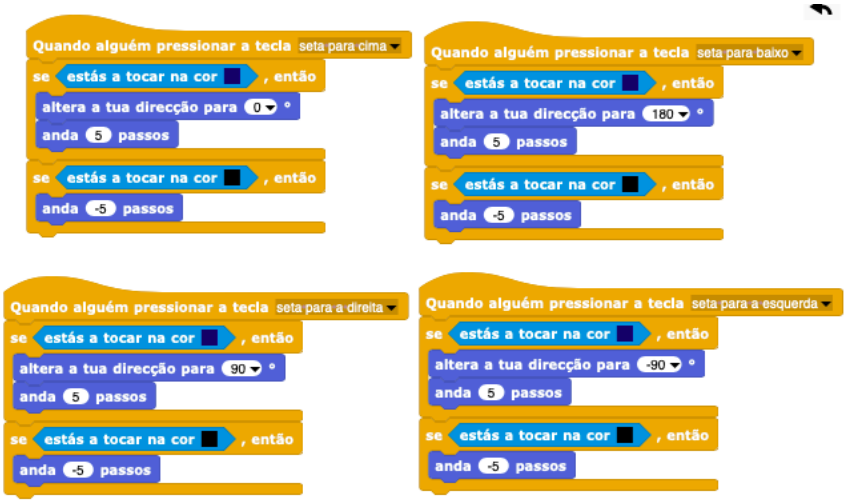
[Passo 2]

Eles têm que desenhar o pacman e o fantasma vermelho. Para a estrela podemos simplificar e desenhar um círculo dentro do Snap!:



[Passo 3]

Para que o pacman se mova, podemos usar diferentes possibilidades. A amostra abaixo é uma delas. Nela, usamos um sistema de eventos para detectar a seta que é pressionada, esquerda, direita, acima ou abaixo. Depois de acontecerem cada um destes eventos, temos que testar se ele está a tocar na cor da área que lhe é permitido mexer-se. Se este for o caso, ele vira-se primeiro para aquela direção e faz a jogada. Mas se ele tocar na cor das paredes, ele tem que voltar atrás, porque de outra forma ele ficaria preso à parede devido à primeira condição.



[Passo 4]

A próxima tarefa é programar as estrelas. As estrelas serão todas as mesmas, mas haverá muitas delas. Neste caso, é melhor fazer um objeto e criar os seus clones do que fazer múltiplos objetos idênticos (no nosso caso 20). No início do jogo o primeiro clone vai aparecer dentro do labirinto de forma aleatória, depois quando o jogador a recolher vai desaparecer e uma nova será criada numa localização aleatória diferente. De forma a criar o primeiro clone no início do jogo colocamos este código num guião de cenário.

De forma a esconder o objeto original e apenas mostrar os clones, temos que fazer isto no início da programação.

De forma a encontrar as localizações aleatórias adequadas temos que observar certas restrições. Se a estrela é criada numa parede, um jogador não consegue alcançá-la, ou seja não a podemos colocar lá. A estratégia para o fazer é a seguinte:

1. Temos que encontrar a posição aleatória x,y para posicionar o clone da estrela. Ambas as coordenadas x e y estão no mesmo intervalo[-140, 140]. Então, escolhemos um número aleatório daquele intervalo para ambos.
2. Em seguidas, verificamos se o clone está a tocar na parede. Neste caso, esta localização não é permitida.
3. Se a localização estiver bem, temos de mostrar o clone (lembre-se que o original está escondido e o clone estaria escondido também se não utilizássemos o bloco mostra-te) e no loop infinito verifique se a colisão com o jogador ocorre.
4. Se a localização não estiver bem, criamos um novo clone (esperando que para o novo clone, os números aleatórios sejam escolhidos para que seja posicionado numa localização permitida) e apagar esta.
5. De forma a contar os clones recolhidos temos que informar a contagem total das estrelas que tem de ser definida fora do clone, ex. no jogador. Isto pode ser feito pela transmissão de uma mensagem que a colisão ocorreu. Depois podemos

apagá-la.

[Passo 5]
Em seguida, nós programamos um fantasma. Ele tem que se movimentar aleatoriamente pelo labirinto e tem que mudar de direção quando encontra com uma parede. No Snap! as direções são expressadas em graus:

1. 0 graus - ACIMA
2. 180 graus - ABAIXO
3. 90 graus - DIREITA
4. 270 graus - ESQUERDA

Em outras palavras, se escolhemos aleatoriamente números entre 0 e 3 e multiplicá-los por 90 cria-se uma direção aleatória!
Ele tem que mover-se até colidir com o pacman. Então o jogo acaba.

[Passo 6]

Agora temos que programar quando o jogador vencer o jogo. Isso acontecerá quando ela coletar 20 estrelas. Nós temos um contador de estrelas dentro do pacman script. No início nós começamos com zero, e aumentamos seu valor por 1 cada vez que que o clone envia a mensagem que o jogador coletou-a. Se o contador chegar a 20, o pacman ganha e temos que parar o jogo.

Instrumentos e recursos para o Professor	• Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_clone • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Recursos/Materiais para os alunos	• Template em Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_template • Instruções para os alunos (C4G21_InstructionsForStudent.docx)

REFERÊNCIAS

Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena.

Rugelj, J. (2019). Game design based learning of programming.

Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.