O3 –Conteúdo de suporte instrucional

O3/A2 - GUIA DE UTILIZADOR SOBRE A ABORDAGEM DE JOGO SÉRIO CODING4GIRLS

Dados do documento

Deliverable: O3/A2 - User guide on the CODING4GIRLS serious game approach

Intellectual Output No - Title: O3 – Instructional Support Content

Intellectual Output Leader: South-West University "Neofit Rilski" (Bulgaria)

Contribuídores

South-West University "Neofit Rilski", Bulgaria

Daniela Tuparova, Boyana Garkova, Ivanichka Nestorova, Rositsa Georgieva

UTH – Greece

Hariklia Tsalapata, Olivier Heidmann, Kostas Katsimentes, Christina-Roxani Taka, Sotiri Evangelou, Nadia Vlachoutsou

University of Rijeka – Croatia

Nataša Hoić-Božić, Martina Holenko Dlab, Ivona Franković, Marina Ivašić Kos

EU-Track – Italy

Michela Tramonti, Alden Meirzhanovich Dochshanov and Luigi Tramonti

Virtual Campus - Portugal

Carlos V. Carvalho, Rita Durão, Carolina Novo, Hanna Schiff, Sandra Cruz, Chiara Zanchetta

University of Ljubljana - Slovenia

Joze Rugelj, Mateja Bevcic, Spela Cerar, Matej Zapushek

GOI - Turkey

Ahu Şimşek, K.Fatih Mutlu, Abdurrahman Saygın

Disclaimer

Este projecto foi financiado pelo Programa Erasmus+ da União Europeia.

ÍNDICE

1.INTRODUÇÃO

O projeto Coding4girls aborda a educação e a formação abertas e inovadoras embutidas na era digital, visando habilidades de programação que estão em alta demanda numa sociedade orientada pela tecnologia. Além disso, a capacidade de projetar algoritmos, codificar e programar tornaram-se habilidades pessoais relevantes, pois estão conectadas com raciocínio lógico e habilidades de resolução de problemas. Como tal, as competências em ciência da computação são altamente desejáveis, uma vez que fazem parte das capacidades exigidas pelos empregadores em sectores orientados para a inovação, que irão impulsionar um crescimento económico sustentável nos próximos anos, conduzindo a um aumento do emprego e, consequentemente, da coesão social.

No entanto, existe, atualmente, uma escassez de profissionais especializados em ciências da computação e as empresas enfrentam dificuldades em encontrar trabalhadores que possuam o conjunto de competências TIC necessárias. A CODING4GIRLS tem como objetivo colmatar esta lacuna, abordando de forma eficaz as habilidades de ciência de computação no final dos anos escolares básicos e nos primeiros anos do ensino médio, que é o momento em que muitos estudantes perdem o interesse em ciências da computação. Além disso, há uma lacuna de género amplamente reconhecida nesta área, ou seja, a maior parte das pessoas do gênero feminino não estão interessadas em seguir uma carreira na área das ciências da computação.

CODING4GIRLS aborda a educação inclusiva, formação e juventude, promovendo oportunidades iguais entre o género feminino e masculino nas carreiras altamente gratificantes de ciência da computação. O projeto responde a este objetivo através da abordagem de equívocos e atitudes dos alunos, professores e pais sobre carreiras de ciência da computação e os seus benefícios para ambos os géneros, demonstrando as suas ligações com a vida real e destacando o facto de que tais carreiras são recompensadoras independentemente do género ou origens; mas também abordando o baixo desempenho em habilidades de ciência da computação, que fazem parte do grupo de competências de ciência e tecnologia mais amplo (STEM).

O projeto ajuda a desenvolver a formação necessária entre os alunos da escola que lhes permitirá ter sucesso nos seus futuros esforços académicos (no ensino superior ou outra educação continuada em ciências da computação) ou profissionais (em atividades profissionais relacionadas com a informática). Mas o CODING4GIRLS aborda, também, o desenvolvimento de competências dos professores e o perfil da profissão docente de forma a capacitar os educadores para, efetivamente, adquirir habilidades desejáveis de ciência da computação entre os seus alunos. Mais ainda, permite que os educadores liderem iniciativas que promovam perspetivas de igualdade de género na procura de caminhos académicos ou profissionais na área da ciência.

Este guia de utilizador para o professor faz parte dos resultados intelectuais desenvolvidos no âmbito do projeto. Os professores encontrarão informações sobre: principais conceitos fundamentados na metodologia Coding4girls – aprendizagem baseada em jogos; jogos sérios; *design thinking;* posicionamento do género feminino sobre as TIC e as suas percepções de jogos sérios; características do ambiente do jogo Coding4girls e com ele os dois componentes – parte do professor e parte do aluno; revisão de abordagens de aprendizagem baseada em jogos para ensinar programação e exemplos de ambientes de programação úteis. O guia lida também com folhas de aprendizagem de lições e a sua adaptação de acordo com a metodologia Coding4girls e ambiente baseado em jogos desenvolvidos.

2. CONCEITOS BÁSICOS

2.1. Jogos Sérios

O conceito de jogo é bastante amplo, o que torna difícil apontar as características mais importantes que fazem de uma determinada atividade um jogo. Para Thorton, a característica mais importante do jogo é a interatividade. Johnston vê o jogo como uma atividade que tem regras que todos os participantes têm que obedecer, um ou vários objetivos, interação e visuais dinâmicos (Johnston & Felix, 1993). Malone(Malone T., 1981) na sua teoria, define elementos como fantasia, curiosidade, desafio e controle como os componentes mais importantes em cada jogo.

A alocação mais compreensiva é feita por Garris et al. (Garris, Ahlers e Driskell, 2002) que alega que os componentes são competição, desafio, interações sociais, conversação e fantasia. A teoria de Prensky da aprendizagem baseada em jogos é uma das mais influentes. O autor argumenta que o jogo consiste nesses elementos: regras, metas e tarefas, resultados e feedback, situações de conflito(competição, desafio e oposição), interação e fantasia(Prensky, 2001). Os autores do livre "Jogos Sérios" definem o jogo como uma atividade voluntária(uma forma de liberdade), que absorve a atenção total do jogador e é jogado de acordo com as regras estabelecidas que todos os jogadores têm de seguir(Michael & Chen, 2006).

Uma das características mais expostas do jogo é a interatividade e refere-se à interação com o computador, oponente (computador ou controlado pelo humano) ou com outros jogadores de equipa. Os jogos de computador são divididos pelo número de jogadores envolvidos simultaneamente num jogo para um único jogador ou *multiplayer.* De acordo com o género, existem jogos de árcade, ação, guerra, *role playing,* estratégia, tabuleiro, desporto, simulação e aventura. Os gráficos que indicam a popularidade dos jogos de computador mostram que os jogos mais populares são os de jogadores múltiplos, estratégia e jogos de ação.

Para realmente compreender o poder do jogo, vamos considerar, primeiramente, uma brincadeira de crianças. As brincadeiras infantis são conhecidas como uma das atividades mais importantes que ajudam a desenvolver habilidades importantes para toda a vida. Ao brincar, a criança está a melhorar as suas capacidades intelectuais à medida que descobre os

primeiros conceitos básicos do mundo real e faz conexões viáveis entre eles. Jean Piaget vê os jogos como a incorporação de novas matérias na estrutura cognitiva existente e na consolidação do comportamento recém-aprendido. Freud enfatiza os benefícios emocionais na brincadeira, argumentando que esta atividade reduz a ansiedade objetiva e instintiva – ajudando a criança a resolver problemas emocionais. O construtivismo social acredita que o jogo é importante devido ao desenvolvimento de habilidades sociais.

Aprender com jogos é uma das atividades mais comuns em idades precoces, mas os efeitos positivos de jogar e dos jogos na aprendizagem são, muitas vezes, ignorados, na educação formal. Muita pesquisa tem sido feita num campo de aprendizagem baseada em jogos, mostrando que se o material de aprendizagem for apresentado em formato de jogo de computador tem efeitos positivos na motivação. Os jogos podem ajudar os alunos a chamar a atenção sobre o material de aprendizagem e, ao mesmo tempo, é uma fonte de diversão que lhes dá prazer. Ter esses dois componentes (diversão e prazer) num processo de aprendizagem, motiva os estudantes que ficam mais dispostos a aprender. Outros benefícios pedagógicos são a aprendizagem colaborativa (capacidades multijogador), a aprendizagem experimental e ativa, a aprendizagem baseada em problemas e atividades da vida real. O uso das TIC é hoje comum numa educação formal, o que nos dá a oportunidade de desenvolver materiais de qualidade, a fim de promover a aprendizagem.

Os jogos sérios geralmente referem-se a jogos usados para formação, publicidade, simulação ou educação. Estes permitem que os alunos experimentem situações que são impossíveis de experienciar no mundo real por várias razões, como segurança, custo ou tempo. Os alunos devem ter resultados de aprendizagem definidos e são reivindicados para ter impactos positivos sobre o desenvolvimento dos jogadores de novos conhecimentos ou habilidades diferentes. Para projetar um bom jogo sério, temos de ser capazes de projetar e produzir um bom software. Porém, jogos sérios são muito mais do que o software. Além disso, também precisamos de ser capazes de projetar e produzir boas instruções. O valor potencial de jogos sérios para a aprendizagem parece ser alto, mas permanece a resistência ao uso dos jogos na sala de aula. Uma maneira razoável de convencer mais professores a experimentar é através de pedagogia, conectando elementos de projetos de jogos existentes com aprendizagem e teorias instrucionais já aceites.

Rieber, Smith e Noah(Rieber, Smith & Noah, 1998) afirmaram que em 1998 já existiam duas aplicações distintas de jogos de educação: jogo e *design* de jogos. Jogar é a abordagem

tradicional onde se fornece jogos prontos para os alunos. O *design* de jogos assume que o ato de construir um jogo é em si mesmo um caminho para a aprendizagem, independentemente de o jogo ser ou não interessante para outras pessoas. A ideia de "aprender criando" baseia-se no pressuposto de que a participação ativa no processo de criação e desenvolvimento é a melhor maneira de aprender algo. Esta abordagem ganhou maior destaque devido à proliferação de ferramentas de *design* e criação baseadas em computador.

2.2. Aprendizagem baseada em jogos

Existem várias razões que chamam a atenção dos educadores para os jogos (Gee, 2007). Na educação formal, experimentamos uma mudança do modelo didático tradicional, que é focado na instrução, para o modelo centrado no aluno que enfatiza o papel do aluno ativo. Também mudamos a visão das metas de aprendizagem de níveis taxonómicos mais baixos, como apenas recordar informações, para níveis mais altos, como encontrar e usar informações em novos ambientes (Rugelj, Serious computer games design for active learning in teacher education, 2016). Prensky (Prensky, 2001), Gee(Gee, 2003) e Whitton(Whitton, 2009) definiram a aprendizagem baseada em jogos como um processo de aprendizagem com recurso aos jogos digitais. Os jogos podem motivar a aprendizagem, aumentando, desta forma, a possibilidade de os resultados desejados da aprendizagem serem alcançados. A aprendizagem é definida como a aquisição de conhecimento ou habilidades através da experiência ou prática, e que melhor maneira de aprender do que através de um jogo (Pivec & Kearney, 2007), (Pivec, Koubek & Dondi, 2004). Quase todos os estudantes sobre a aprendizagem baseada em jogos mostram que os alunos são altamente motivados quando os materiais de aprendizagem são apresentados num formato de jogo de computador e isso terá efeitos positivos no processo de aprendizagem (Zapusek & Rugeli, 2013). Os alunos precisam de motivação para se concentrarem no que precisam de aprender, mas para qualquer aprendizagem de qualidade ocorra isto não é suficiente. Comparar os resultados de aprendizagem dos alunos que aprenderam com jogos de computador e aqueles que aprenderam com outro tipo de material de aprendizagem mostra que não há diferença significativa entre eles. Isto geralmente acontece devido ao *design* inadequado do jogo. Os jogos podem ser muito atraentes para os alunos, mas apenas se entreter e não ensinar, o uso dos jogos na educação não faz muito sentido. Então, temos que descobrir quais são os elementos que tornam um jogo de computador num jogo de computador educacional.

Gross (Gross, 2003) afirma que os jogos digitais para fins educativos devem ter objetivos de aprendizagem para aumentar as habilidades cognitivas e intelectuais dos alunos.

De acordo com Malone(Malone T. W., 1981b) e Garris et al. (Garris, Ahlers & Driskell, 2002), os elementos que contribuem para os valores educacionais dos jogos digitais são estímulos sensitivos(representações visuais e de áudio do material de aprendizagem), fantasia (contexto apresentado num cenário imaginário), desafio(situação exigente ou estimulante) e curiosidade (desejo de conhecer ou aprender). Estes elementos devem ser incorporados numa plataforma integrada, para estruturar objetivos e regras, um contexto de aprendizagem significativa, uma história atraente, feedback imediato, um elevado nível de interatividade, desafio e concorrência, elementos aleatórios de surpresa, e ambientes ricos para a aprendizagem.

Um jogo pode ser conduzido para a aprendizagem, uma vez que envolve estimulação mental (e às vezes física) e desenvolve habilidades práticas- obriga o jogador a decidir, escolher, definir, prioridades, resolver problemas, etc. Recompensa imediata (e feedback) é um fator de motivação importante, se é traduzido como entidades de jogo (mais poder de vida, acesso a novos níveis, etc.) ou como impulsos neurológicos (felicidade, sentimento de realização, etc.). Os jogos podem ser ambientes sociais, às vezes, envolvendo grandes comunidades distribuídas. Estes implicam (Baptista & Vaz de Carvalho, 2010) capacidades de auto-aprendizagem (os jogadores são muitas vezes obrigados a procurar informações para dominar o jogo em si), permitem a transferência de aprendizagem de outras realidades e são inerentemente experienciais como o envolvimento de múltiplos sentidos.

Van Eck (van Eck, 2006) argumenta que jogos e brincadeiras podem ser ambientes de aprendizagem eficazes, não porque eles são divertidos, mas porque eles são imersivos, exigem que o jogador tome decisões importantes frequentes, tenha objetivos inteligentes, e se adapte a cada jogador individualmente e se envolva nas redes sociais.

Se considerarmos um modelo de aprendizagem baseado em jogos por Garris, Ahlers e Driskell (Garris, Ahlers & Driskell, 2002), a principal característica do jogo educativo é que o conteúdo instrutivo é difundido dentro das características do jogo. Os alunos estão a divertir-se a jogar, esquecendo a parte da "aprendizagem" da experiência, mesmo que neles estejam a ser constantemente apresentados com novos conceitos que eles têm que se adaptar, a fim de serem bem-sucedidos no jogo. Devemos promover a motivação com o *design* do jogo, impulsionando a repetição dos ciclos dentro do contexto do jogo. Espera-se que o jogador

elicie comportamentos desejáveis com base em reações emocionais e cognitivas que resultam da interação e do feedback do jogo.

Gee (Gee, 2003) argumenta que os recursos a jogos eletrónicos com alto potencial de aprendizagem se enquadram em duas categorias: características *"non-game"* , que podem aparecer em contextos não-jogo, e característica do jogo que se relacionam mais com "*gameness"* dos jogos. Apesar desta distinção, não deve ser assumido que as características de um não-jogo funcionariam também para a aprendizagem se elas fossem retiradas do jogo.

As características "não-jogo" de jogos com alto potencial de aprendizagem são:

- Empatia por sistemas complexos - olhar o sistema complexo de dentro para entender como suas variáveis estão a interagir.
- Simulações de experiência e preparativos para a ação.

  Nos jogos eletrónicos, os jogadores vêem o mundo virtual em termos de como ele proporciona os tipos de ações que precisam tomar para atingir o objetivo de vencer.
- Inteligência distribuída através da criação de uma ferramenta inteligente.

  Bons jogos de vídeo distribuem inteligência entre uma pessoa do mundo real e personagens virtuais artificialmente inteligentes, a fim de representar uma macro e micro visão da situação.
- Trabalho de equipa multifuncional

  Bons jogos de vídeo podem ser capazes de ensinar colaboração e trabalho em equipa multifuncional para instituições como escolas e locais de trabalho. Os jogadores interagem uns com os outros não em termos das suas características do mundo real, mas, através das duas identidades de jogos funcionais. Eles também podem optar por usar a sua raça, classe, cultura e género do mundo real como recursos estratégicos, porém, eles não são forçados a ter características raciais, de género, culturais ou de classe pré-definidas.
- Significado situado

  O diálogo e a experiência são essenciais para que as pessoas possam relacionar palavras com ações reais, funções e resolução de problemas. Uma vez que os jogos de vídeo são simulações de experiências, eles podem situar a linguagem em contextos específicos.

- *"Open-endedness"* : fundir o pessoal e o social

  Em bons jogos abertos, os jogadores constroem os seus próprios objetivos, que são baseados em seus próprios desejos, estilos e origens. A combinação de objetivos pessoais e em jogo produz um estado de alta motivação.

As características de um "bom jogo" que se relaciona com a aprendizagem eficaz são as seguintes:

- Motivação

  Os jogos de vídeo são profundamente motivadores para os jogadores, e é importante entender as fontes dessa motivação para fornecer uma base para a aprendizagem.
- O papel do fracasso

  O preço da falha é desvalorizado e é visto frequentemente como uma maneira de aprender o teste padrão subjacente. Essas características de falha nos jogos permitem que os jogadores assumam riscos que podem ser muito caros.
- Competição e colaboração

  Muitos jogadores, incluindo os jovens, gostam de competir com outros jogadores em jogos, mas podem não ver a competição como agradável e motivadora na escola. A competição em jogos eletrónicos é vista pelos jogadores como social, tanto sobre jogos quanto sobre ganhar e perder.

O design de jogos que se relaciona com:

- Interatividade - jogador não só passivamente consome conhecimento, mas tem controle sobre o conteúdo;
- Personalização - baseada em estilos de aprendizagem e proporcionando múltiplas rotas para o sucesso;
- Identidades fortes - Bons jogos oferecem aos jogadores identidades que desencadeiam um investimento profundo por parte do jogador e que estão claramente associados às funções, habilidades e objetivos que cada um tem que realizar no mundo virtual;
- Problemas bem sequenciados - Em abordagens conexionistas à aprendizagem, argumenta-se que o sequenciamento é crucial para a aprendizagem eficaz em domínios complexos;

- Um nível agradável de frustração - ajustar os desafios de tal forma que uma variedade de jogadores possa experimentar o jogo como desafiador, mas capaz;
- Um ciclo de perícia - ciclos repetidos de prática estendida e testes de domínio;
- *"Deep*" e *"fair"* - o jogo deve ser desafiador, mas configurado de uma forma que leva ao sucesso.

Os elementos do jogo devem ser, inicialmente, simples e fáceis de se aprender e indo-se tornando mais complexos à medida que o jogador os vai dominando.
Os jogos de computador usados na educação estão provados que aumentam a motivação. Porém, alunos altamente motivados não são suficientes para uma aprendizagem ocorrer. Também são necessários bons materiais de aprendizagem para que os alunos realmente ganhem novos conhecimentos a partir de materiais apresentados em forma de um jogo de computador.
Apesar da influência positiva do jogo na motivação, os professores ainda hesitam em utilizar jogos sérios no ensino. Como eles apontaram na nossa investigação, os jogos podem consumir muito tempo para serem usados numa sala de aula. Portanto, eles, muitas vezes, usam jogos de aprendizagem séria como uma atividade de aprendizagem em casa ou como uma motivação nas aulas introdutórias.

2.3. Metodologia do "Design thinking"

As ideias de *"design thinking"* surgiram no final da década de 1960, mas como um conceito *"design thinking"* emergiu na Universidade de Stanford no final dos anos 1980 e 1990. No mundo dos negócios, o método generalizou-se depois de ter sido promovido por Tim Brown da IDEO, que afirmou: "Design *Thinking* é uma abordagem centrado no ser humano para a inovação, baseia-se nas ferramentas do design para integrar as necessidades das pessoas, capacidades tecnológicas e requisitos de sucesso empresarial" (IDEO Design thinking, 2008).
No início do século XXI, o *design thinking* tornou-se extremamente popular e as empresas de TI começaram a aplicar esta abordagem para desenvolver os seus produtos. Desde 2005, por exemplo, a SAP, tem vindo a aplicar o *design thinking* "como uma filosofia de resolução de problemas e como uma abordagem inovadora focada no utilizador final" (Innovation Starter, 2015), e como resultado, o desenvolvimento de novos produtos conduz ao resultado

desejado. Empresas como P&G, IBM Design, Google, Amazon e Cisco integram o "design thinking" em todo o seu negócio e organização (Naiman 2019).

O conceito desenvolvido pela Universidade de Stanford identifica o "*design thinking"* como uma metodologia líder para a resolução criativa de problemas e a criação de inovação e é usado diariamente por milhares de empresas e organizações em todo o mundo. O objetivo é desenvolver em adolescentes as habilidades do século XXI- trabalho em equipa, resolução de problemas, criatividade, empatia, confiança, paciência, concentração, experimentação." (Karakehayova, 2019).

O modelo Stanford é composto por 5 fases (Fig. 2.1).

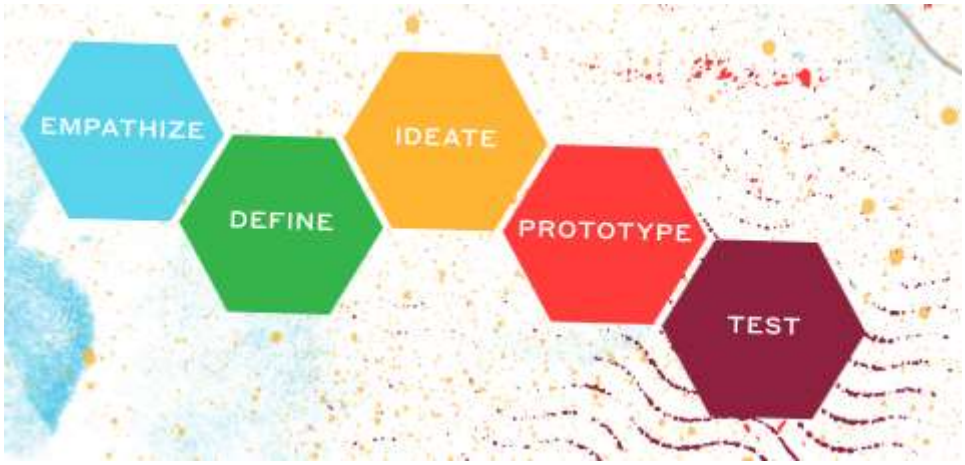


Fig. 2.1. Fases no modelo de "*design thinking*" de Stanford (Doorley, Holcomb, Klebahn, Segovia, & Utley, 2018)

As principais fases e características do *design thinking* podem ser resumidas da seguinte forma: (Tabela 2.1.) *(Georgieva & Tuparova, 2020)*

Table 2.1. Fases e características do design thinking *(baseado no (Innovation Starter, 2015), (Naiman, 2019), (A project of the K-12 Initiative at Stanford University, 2009)*

Práticas	Príncipios	Fases		
		Processo curto	Processo alargado	dSchool Stanford
Desenvolver uma	Descoberta-	Observar e estudar o		Ter

compreensão profundamente empática das necessidades do usuário	Enfrentar um novo desafio. Como resolvê-lo?	comportamento e a experiência dos usuários, pesquisando e definindo um problema e um ponto de vista através da empatia.		empatia
Formação de equipas heterogéneas	Interpretação - O que aprendi e como interpretá-lo?	Gerar muitas ideias num curto período de tempo.		Definir
Conversas baseadas em diálogo	Ideação - Eu vejo uma oportunidade, como moldá-la em uma ideia?	Selecção e classificação de ideias		Idealizar
Gerar decisões através da experimentação;	Experimentação - Eu tenho uma ideia de como construí-lo?	Prototipagem - um produto que recebe feedback rápido dos usuários.		Prototipar
Usar um processo estruturado e facilitado.	Evolução - Eu tentei algo novo, como vou desenvolvê-lo?	Testes/Feedback - como melhorar o protótipo, se necessário		Testar
			Testar se a ideia funciona	
			Aprendizagem e melhoria com base no feedback	

O *design thinking* não só beneficia o negócio, mas também os setores não governamentais, educacionais e da saúde. Cada vez mais e mais instituições educacionais estão a utilizar o

design thinking. Gradualmente, as boas práticas para a aplicação do *design thinking* na educação estão a aumentar. (Georgieva &Tupsrova, 2020). Na implementação da abordagem de *design thinking*, o professor necessita estabelecer uma conexão entre as fases individuais do modelo design thinking e os métodos de ensino apropriados que podem ser aplicados. O design thinking requer a aplicação de uma ampla gama de métodos de ensino.

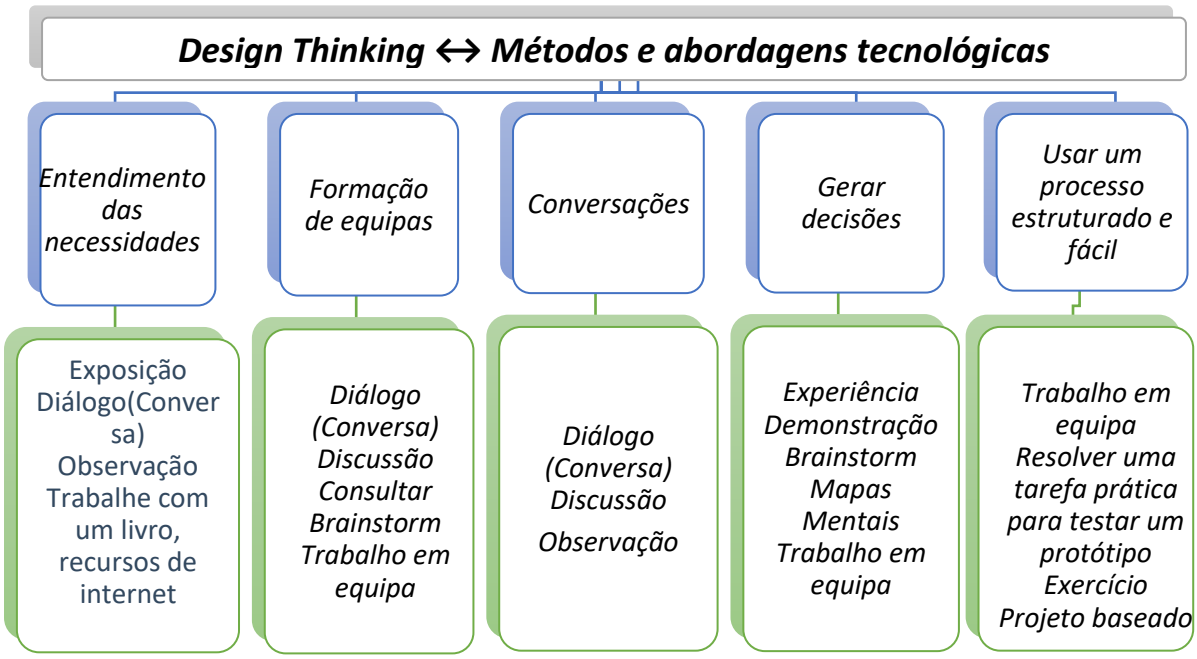


Fig. 2.2. Métodos de ensino e design thinking (Georgieva & Tuparova, 2020)

2.4. Teorias de aprendizagem e aprendizagem baseada em jogos

Muitos jogos de computador educativos foram projetados de acordo com a teoria comportamental da aprendizagem no passado (Rugelj & Lapina, 2019). Estes foram implementados na forma de instrução programada. Foi oferecido aos alunos um estímulo na forma de uma pergunta ou qualquer outro tipo de tarefa ou problema que precisa ser resolvido. Os alunos respondem imediatamente, selecionando uma das respostas disponíveis. Se a resposta estiver correta, o jogo fornece algum tipo de resposta positiva em forma de uma reação positiva da personagem ou uma melodia feliz que estimula emoções positivas. Este exemplo de ação-reação reforça a conexão entre uma pergunta e uma resposta correta. No caso de uma resposta errada, uma reação é fornecida em forma de estímulos negativos e a conexão é enfraquecida. Jogos de apontar e clicar e quizzes têm o

conceito de exercício e prática construído e são representantes típicos de jogos com base no comportamento. Estes são adequados para aprender operações aritméticas básicas ou para apoiar a memorização de dados fractográficos, por exemplo, objetivos de aprendizagem nos níveis mais baixos da taxonomia de Bloom.

A teoria de aprendizagem cognitiva enfatiza a atividade cognitiva do aluno e a formação de modelos mentais apropriados. Os alunos aprendem conceitos fundamentais com o seu professor ou formam recursos de aprendizagem e, em seguida, usa a dedução lógica para adquirir novos conhecimentos. Puzzles e jogos de estratégia como um ambiente para a tomada de decisão são exemplos da abordagem cognitivista.

As formas mais avançadas de jogos baseado em teoria cognitiva têm por base sistemas de tutoria inteligente, onde os algoritmos de aprendizagem de máquina são introduzidos para modelar o conhecimento do estudante e do especialista, visando fornecer material de aprendizagem personalizado.

O construtivismo é uma visão alternativa que sugere que os alunos construam o seu próprio conhecimento; uma série de representações de conhecimentos construídas individualmente, todas igualmente válidas. A aprendizagem é um processo ativo de construção (em vez de aquisição de conhecimento), construído recursivamente sobre o conhecimento que o usuário já tem. Num processo de construção, os dados sensoriais são combinados com os conhecimentos existentes para criar novos mentais viáveis, que são, por sua vez, a base para a construção posterior.

A aprendizagem construtivista enfatiza a aprendizagem da descoberta e da investigação, argumentando que os alunos devem ser colocados num ambiente (que pode ser modelado com um jogo de computador) onde eles constroem o seu próprio conhecimento. Três princípios fundamentais que definem a visão construtivista de aprendizagem:

- cada pessoa forma a sua própria representação do conhecimento,
- a aprendizagem ocorre quando a exploração dos alunos revela uma inconsistência entre a sua representação atual de conhecimento e a sua experiência,
- a aprendizagem ocorre num contexto social e a interação entre os alunos e os seus pares é uma parte necessária do processo de aprendizagem.

Os materiais de aprendizagem fornecem instruções que consistem em apoiar a construção do conhecimento, em vez de declarar o conhecimento de forma comportamental. Simulações de jogos de computador replicam vários cenários da vida real em formato de jogo

de computador. Apresentam um modelo de realidade abstrata em que o aluno vive num determinado papel. A tarefa do professor é fornecer orientação e feedback quando o aluno está a aprender, isto é, construir modelos mentais viáveis.

Os jogos podem conduzir a mudanças nas atitudes, comportamento e habilidades do jogador. Shute e Ke (Shute & Ke, Games, Learning and Assessment, 2012) descobriram que existe uma convergência entre os elementos centrais de um bom jogo e as características da aprendizagem produtiva. Os designers de jogos têm muito para nos ensinar sobre a aprendizagem, e a teoria da aprendizagem contemporânea podemos ensinar como projetar melhores jogos (Shute, Rieber & Van Eck, 2012). Marshall Mcluhan, filósofo canadense da teoria da comunicação, que previu a World Wide Web nos anos sessenta do século anterior, quando falou sobre a aldeia global, afirmou: "Qualquer um que faz uma distinção entre jogos e aprendizagem não sabe a primeira coisa sobre um ou outro" (Shute, Rieber & Van Eck, 2012).

Whitton e Moseley (Whitton & Mosely, 2012) propuseram uma estrutura para boas práticas de design de jogos sérios, partindo de uma perspetiva de aprendizagem ativa. De acordo com as suas orientações, o ambiente de jogo deve apoiar a aprendizagem ativa, incentivando a exploração, a resolução de problemas e a investigação, gerando um envolvimento com objetivos explícitos e exequíveis, ser adequado ao contexto de aprendizagem, apoiar e proporcionar igualdade de oportunidades a todos os alunos, prestar apoio contínuo com uma introdução gradual de complexidade crescente, apoiado com ajudas e sugestões.

Vários estudos têm demonstrado que os jogos sérios podem apoiar a aprendizagem com motivação, envolvimento e diversão (Hijo-Neira, Velázquez-Iturbide, Pizarro-Romero & Carrico, 2015).

Aprender programação requer muitas competências, tais como pensamento lógico, resolução de problemas e a capacidade de entender conceitos abstratos. Por esta razão, muitos alunos consideram a programação de computador uma disciplina difícil de aprender. Este facto, pode conduzir a uma baixa motivação para estudar cursos introdutórios de programação. A fim de melhorar a motivação e a atitude de aprendizagem dos alunos em relação à programação, os professores estão à procura de abordagens simulativas para a aprendizagem.

A programação é mais facilmente aprendida através da prática e, se os alunos devem aprender de forma eficaz, pelo menos parte dessa prática terá que ser autodirigida ou em

colaboração com os colegas. O papel fundamental do professor é persuadir os alunos a fazer isso e, desta forma, motivá-los (Feldgen & Clua, 2004).

Os alunos do curso introdutório de programação computacional desenham e desenvolvem programas (Rugelj & Lapina, 2019). Isto é típico de uma abordagem de aprendizagem ativa. Se introduzirmos o trabalho de projetos em grupos, o que tem provado ser uma maneira mais eficaz de aprender programação (Nancovska, Serbec, Kaucic & Rugelj, 2008), podemos realmente falar sobre o princípio da aprendizagem trialógica. O plano de aprendizagem triológica consiste em formas de aprendizagem em que os alunos de forma colaborativa, desenvolvem, alteram ou criam um artefacto comum (isto é, no nosso caso, um programa de computador) num processo sistemático (Kafai, 1995). Foca-se na interação que ocorre com a criação de artefactos concretos, não apenas entre pessoas (“abordagem dialógica), ou dentro da mente (“abordagem monológica”) (Paavola &Hakkarainen, 2005).

3. RAPARIGAS, TIC E JOGOS SÉRIOS

3.1. Posição das mulheres no sector das TIC - tecnologia de ponta

A sociedade digital cria demandas crescentes nas competências dos povos modernos. Existe uma necessidade crescente de pessoas que não só são digitalmente alfabetizadas, como também têm as habilidades para criar e manter aplicativos de software em vários campos da atividade humana. Os meios de comunicação social comentam, constantemente, a falta de especialistas qualificados no sector das tecnologias da informação e comunicação (TIC). Há um crescimento contínuo deste setor.

Em 2019, cerca de 7,8 milhões de pessoas trabalharam como especialistas em TIC em toda a União Europeia (UE). (Eurostat, 2020). Porém, estes números representam apenas 3,9% de todos os trabalhadores na UE.

De acordo com (Eurostat, 2020) na UE-27 em 2011, as mulheres no setor da TI eram cerca de 17% e em 2019 17,9%. Segundo os dados do Eurostat para o período 2007-2019, em média, para os 27 Estados-Membros da UE, o número de mulheres no sector das TI diminuiu de 22,5% para 17,9%. Em 2019, as mulheres representaram 17,3% do número total de pessoas empregadas na UE com uma educação em TIC, face a 20,2% em 2009. (Eurostat, 2020).

O estado atual do número de trabalhadoras mulheres no setor de TIC nos países parceiros do projeto Coding4girls é apresentado na Fig.3.1.

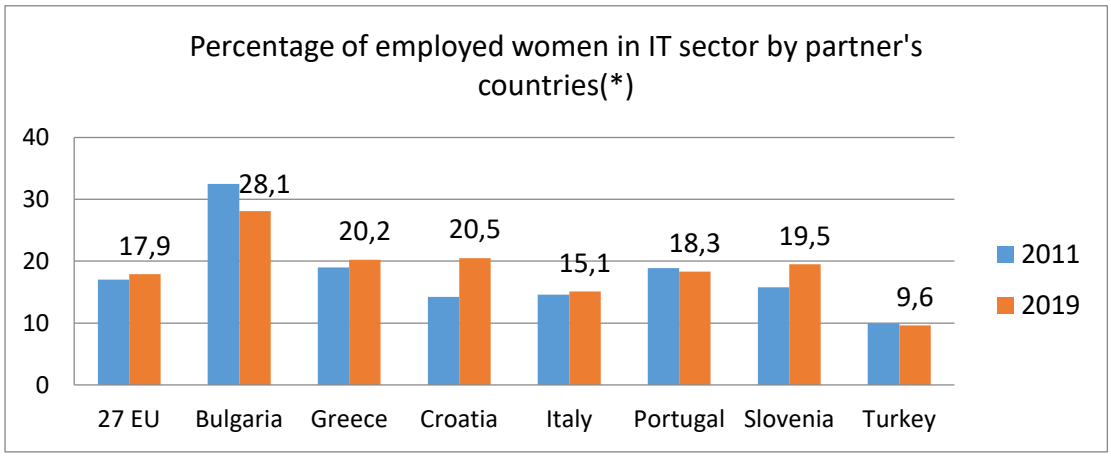


Fig. 3.1. Trabalhadoras mulheres no sector TIC (em percentagem) em 2011 e 2019

3.2. Atitudes das raparigas a jogos sérios

Os resultados do estudo realizado por Vermeulen et al. (Vermeulen, Van looy, Courtois & De Grove, 2011) demonstraram que as diferenças de género estavam consistentemente

presentes, porém a experiência anterior afeta substancialmente as descobertas (Rugelj & Lapina, 2019). As mulheres estão a jogar com mais frequência, mas, em períodos mais curtos de tempo do que os homens. Elas preferem jogos abstratos, curtos e fáceis de dominar, como jogos casuais (ex.Tretis) e jogos nas redes sociais, enquanto, os homens são mais propensos a jogar géneros " núcleos". Esta categoria refere-se a jogos baseados em habilidades, demorados e, normalmente, apresentam gráficos tridimensionais de alta qualidade, como luta, ação-aventura, desportos, corrida, estratégia, RPG e jogos MMO. Outros géneros incluem plataformas, aventura, simulação, festa, jogos sérios, clássicos e casuais. Géneros centrais são jogados mais por homens do que por mulheres. O estudo descobriu que as mulheres gostam de jogos quebra-cabeça, enquanto corridas, ritmo, simulação e jogos virtuais são jogados tanto por mulheres como por homens. Outro estudo descobriu, ainda, que as mulheres preferem jogos de festa(como música e dança) e jogos clássicos.

Alserri et al (Alserri, Zin & Wook, 2018) fizeram uma extensa pesquisa de artigo relacionado sobre elementos eficazes dos jogos sérios, como elementos motivacionais de jogadores de jogos digitais, elementos educacionais eficazes e elementos de preferência feminina. Os autores usaram os resultados para criar um modelo conceitual para o envolvimento de jogos sérios baseados em género. Usamos este modelo no projeto Coding4girls para aumentar a motivação das raparigas para a programação através da seleção apropriada de jogos que serão projetados e implementados pelos alunos na aprendizagem baseada em design de jogos.

Os mesmos autores afirmaram que o estudo revelou que a motivação para jogar um tipo específico de jogo digital depende do género. O estereótipo de que as mulheres não jogam jogos de computador não é válido atualmente. As diferenças nas preferências de género para jogos digitais, encontradas nesta pesquisa, são muito semelhantes às diferenças já apresentadas, identificadas por Vermeulen et al. (Vermeulen, Van Looy, Courtois & De Grove, 2011). As mulheres preferem os jogos mais exploratórios e criativos do que os homens. Elas jogam com mais frequência, contudo por menos tempo do que os homens, e as suas preferências para os géneros de jogos também são diferentes. Verificou-se, no estudo, que o género masculino passa mais tempo a jogar jogos computador do que o género feminino. As mulheres preferem, também, jogos de quebra-cabeça, jogos sociais com recompensas oferecidas nos jogos, jogos educacionais, jogos de simulação e jogos colaborativos e de

exploração, vida virtual, mundo virtual e jogos de festa. Além disso, gostam de jogar jogos de aventura, no entanto, preferem observar os outros a jogar antes de jogá-los. Os elementos de preferência de motivação em jogos para mulheres são: desafio, escapismo, diversão, interação social, fantasia, competição e excitação. Ambos os géneros gostam de jogos de corrida, simulação e realidade virtual.

Phan et al. (Phan, Jardina, Hoyle, & Chaparro, 2012) chegaram a conclusões muito semelhantes nos seus estudos sobre as diferenças de género entre jogadores masculinos e femininos em termos de uso de jogos de computador, preferência e comportamento.

Segundo o estudo de Tuparova (Tuparova, Tuparov & Veleva, 2019 b), existem diferenças nas preferências de rapazes e raparigas nas características e elementos de jogos como: design gráfico; som; possibilidade de jogar o jogo em diferentes dispositivos (computador, tablet, smartphone, etc,; possibilidade de envolver muitos jogadores; possibilidade de jogar on-line; possibilidade de mudar para níveis de jogo mais complexos. A característica dos jogos preferida pelas raparigas é a possibilidade de mudar para níveis de jogo mais complexos. Para os rapazes, os jogos prediletos são os de lógica e enredo do jogo. A característica de menor interesse nos jogos para as raparigas é o som, enquanto que para os rapazes são os prémios nos jogos. Em relação ao dispositivo usado, o smartphone é o mais usado tanto por rapazes como raparigas. Já o segundo e terceiro preferido pelas raparigas é o computador e o tablet respetivamente. Os autores observaram que os rapazes mostram maior preferência do que as raparigas para jogos de história e as raparigas maior preferência para jogos de memória. Independente do género as preferências são para os seguintes tipos de jogos: jogos com elementos sociais; atenção/ concentração, jogos de velocidade de reações, jogos de papel, quebra-cabeças, jogos de aventura. O tipo de jogo de menor interesse para ambos os géneros é o jogo de quebra-cabeça.

Temos que ter em conta a situação específica do projeto Coding4girls, que é realmente fundamentado em aprendizagem baseada em design de jogos e preparação de tarefas interessantes e motivadoras para as raparigas. Um ambiente de programação que apresenta a mesma como um meio para criar filmes animados (isto é, *storytelling*) ou jogos simples que podem ser adequados para raparigas e como a maioria delas pode vir com uma ideia para uma história ou jogo simples que elas gostariam de criar (Wolz, Barnes & Bayliss, 2009). Ambos são naturalmente sequenciais e não são suscetíveis de exigir conceitos de programação avançada imediatamente, os jogos são uma forma de auto-expressão e de

proporcionar às raparigas uma oportunidade de experimentar com diferentes papéis, e amigos não-programadores podem facilmente entender e apreciar uma história animada ou jogo, que fornecem uma oportunidade de feedback positivo. Kelleher (Kelleher, Paush& Kiesler, 2007) usou o "Storytelling Alice" e "Generic Alice" para ambos os casos respetivamente.. Eles afirmam que vários estudos de programados infantis descobriram que quando as raparigas e os rapazes têm uma experiência semelhante com programação de computador, eles ficam igualmente interessados e são eficazes a aprender a programar. Contudo, o desempenho na programação é correlacionado com a quantidade de tempo que os usuários passam a programar e a sua experiência de programação prévia.

4. ENSINAR PROGRAMAÇÃO ATRAVÉS DE JOGOS

Esta seção será focada em apresentar e comparar diferentes abordagens e plataformas/ambientes de jogos que podem ser usados para ensinar habilidades de programação às crianças.

4.1.Abordagens para ensinar programação através de jogos

Há atualmente uma grande variedade de abordagens para ensinar programação para programadores novos.

As abordagens podem ser classificadas como:

- Aprender programação através do design baseado em jogos - também conhecido como aprendizagem baseada em design de jogos - corresponde ao uso de linguagens de programação para aprender a codificar, projetando e criando jogos;
- Aprender programação num ambiente baseado em jogos.
- A metodologia Coding4girls pertence a esta categoria de ambientes de aprendizagem de programação.

Um exemplo é o uso de linguagens de programação visual baseadas em blocos que são, especialmente fáceis de usar e operar (numa base de arrastar e soltar), evitar erros em relação à sintaxe e lógica(Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015) e apresentar conteúdo de uma forma mais intuitiva. Scratch (Resnick, et al., 2009), https://scratch.mit.edu/ , Snap! (Weintrop & Wilensky, 2015) https://snap.berkeley.edu/ , or Alice 2/3 https://www.alice.org/about/ são exemplos bem conhecidos e bem-sucedidos de linguagens de programação visual. De seguida, apresentamos uma tabela comparando as características das mais conhecidas linguagens de programação visual (e ambientes).

Tabela 4.1. Características de alguns ambientes de programação baseados em blocos visuais.

	PLATFORM	TARGET GROUP	LANGUAGE(S)	FREE/PAID
Scratch	Baseado na Web (mas com versão offline)	8-16	Linguagem de programação de computador visual	Gratuito
Snap!	Baseado na	12-20	Linguagem de	Gratuito

	Web (mas com versão offline)		programação de computador visual	
Alice	Windows, Mac OS X, Linux	12-20	Linguagem de programação de computador visual	Gratuito
Tynker	Baseado na Web, iOS	7+	Linguagem visual de programação computacional, JavaScript, Python	Pago

Os jogos de computador também podem ser usados para aprender a codificar, seja encorajando os alunos a desenvolver e criar os seus próprios jogos de vídeo (aprendizagem baseada no game-design) ou permitindo-lhes jogar vários jogos sérios cujos resultados de aprendizagem englobam a programação (aprendizagem baseada no jogo). Os jogos de computador destinados a atividades educacionais têm o potencial de criar um ambiente de aprendizagem motivador e divertido, já que contêm atividades que atendem aos padrões educacionais, objetivos de aprendizagem, fornecem feedback e podem alcançar resultados educacionais elevados (Georgieva & Tuparova, 2019).

A ideia por detrás do uso de jogos para ensinar programação vem do facto de estes, por serem mais envolventes e motivacionais, permitem que os alunos aprendam o pensamento computacional e habilidades de programação em ambientes divertidos e familiares, antes de transferir essas habilidades para a aprendizagem de uma linguagem de programação (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012), 2012). Segundo Kazimoglu et al. (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012), os jogos sérios para aprender programação e pensamento computacional não devem ser criados apenas por diversão, devem, também, considerar um currículo e habilidades, fazendo a diferenças entre várias construções de programação e incentivado boas práticas de programação (por meio de gamificação, como alcançar uma pontuação alta).

Os jogos sérios também podem ser úteis para a programação, transformando operações desagradáveis em experiências interessantes- Shabanah et al (Shabanah, Chen, Wechsler, Carr, & Wegman, 2010) criou um Designer de um Jogo Algoritmo destinado a facilitar a

criação de jogos de algoritmo, a fim de melhorar o envolvimento em sistemas de visualização de algoritmos.

Grivokostopoulou et al (Grivokostopoulou, Perikos, & Hatzilygeroudis, 2016) desenvolveu um jogo para ensinar algoritmos de IA, baseado no jogo Pacman, para que através de visualizações de algoritmos e animações os alunos pudessem ser ajudados no seu processo de aprendizagem, demonstrando a forma como um algoritmo funciona, as suas funções e como tomar decisões adequadas com base em parâmetros específicos.

Com o desenvolvimento generalizado da adoção de velocidades mais rápidas de conexão à Internet e a crescente disponibilidade de computadores nas casas de tosos, plataformas on-line baseadas em jogos como CodeCombat (https://codecombat.com/) e LightBot (https://lightbot.com/) começaram a aparecer. Combéfis et al. (Combéfis, Beresnevičius, & Dagiene, 2016) revisaram os principais tipos de plataformas online usadas para ensinar habilidades de programação e, concluíram que os seguintes fatores podem tornar um jogo sério mais motivador e bem sucedido : 1) um processo de feedback e avaliação; 2) estética; 3) aspetos de colaboração e multijogador; 4) orientação através do jogo; 5) sem consequências negativas; 6) música; 7) um nível médio desafiante para manter o interesse dos jogadores.

Em relação aos resultados de aprendizagem dos jogos sério para a aprendizagem de programação, de acordo com uma extensa revisão de 49 jogos de programação séria por Miljanovic et al (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018), a maior parte dos jogos sérios para a programação têm como foco a resolução de problemas e conceitos fundamentais de programação, com alguns jogos focando-se em estruturas de dados, métodos de desenvolvimento e design de software.

Existem também jogos que, embora não possam ensinar habilidades de programação, são capazes de ensinar de forma indireta, porém eficaz, as habilidades de pensamento computacional chave, como abstração (por exemplo, Tetris), decomposição (por exemplo, Sudoku), pensamento algoritmo(por exemplo, jogos de quebra-cabeças), avaliação (ex. jogos de memória), e generalização(ex. Jogos de padrão) (Frankovic, Hoic-Bozic , & Nacinovic-Prskalo, 2018).

Na aprendizagem baseada no design de jogos, muitas vezes, temos a combinação do uso de linguagens de programação visual e os seus ambientes baseados em blocos com o processo de design e codificação de jogos. Num estudo que comparou o uso de ambientes de

codificação, usando o Scratch e Pascal (linguagem processual através da linha de comando ou editor de texto) com programadores novos, descobriu-se que os alunos estavam muito mais motivados a continuar os seus estudos em ciências da computação ao usar Scratch (Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015). O estudo descobriu, também, que a criação de jogos e histórias torna os alunos mais criativos e autónomos enquanto aprendem programação. Outro estudo, realizado por Weintrop et al (Weintrop & Wilensky, 2015), comparando o uso do Snap! e Java por alunos do ensino médio descobriram que a abordagem baseada em blocos tinha uma curva de aprendizagem mais fácil do que Java - portanto, está de acordo com a visão de que a programação baseada em blocos (como Scratch e Snap!) é mais acessível para programadores novos. De acordo com as descobertas do estudo, isto deve-se ao fato de os blocos serem mais fáceis de ler, devido à natureza visual dos blocos que fornecem dicas sobre como podem ser usados, são mais fáceis de compor e servem como auxiliares de memória.

A abordagem integrada para o ensino de programação e influencia as realizações de alunos de 14 anos é discutida em (Tuparova, Nikolova, & Tuparova, 2020). Esta abordagem combina duas fases:

- "Uso de jogos de computador educativos existentes: para aquisição de conhecimentos e habilidades, motivação de atividades de aprendizagem, desenvolvimento de pensamento algorítmico; para experimentar com jogos de computador educativos existentes para inquirir de regras de jogo, elementos de interface e suas propriedades e eventos;
- Concepção e desenvolvimento de jogos de computador educativos, incluindo o modelo matemático do jogo, design de interface gráfica do usuário (GUI); codificação, testes e verificação."

Para a implementação da abordagem é utilizado o ambiente de programação C #. O modelo é testado com 126 alunos com especialização em Informática, divididos em dois grupos - Experimental e Controle. As realizações dos alunos são medidas por meio de tarefas práticas e testes em papel. Os resultados mostram que não há diferença entre as realizações de rapazes e raparigas no grupo experimental. Mas os autores descobriram que as raparigas no grupo experimental alcançaram resultados mais elevados do que as raparigas no grupo de controlo.

4.2. Ambientes baseados em jogos para ensinar programação

Em seguida, apresentamos alguns exemplos do uso de diferentes ambientes que usam jogos para ensinar codificação e programação.

4.2.1. Aprendizagem pelo design de jogos

Scratch

*Fig.4. SEQ Picture * ARABIC 1* Sample Scratch Script

O Scratch (https://scratch.mit.edu/) é um ambiente de programação baseado em blocos, ou seja, permite que os alunos construam programas juntando blocos de código e recebimento visual (Weintrop & Wilensky, 2015).Desta forma, os alunos familiarizaram-se com os fundamentos da programação sem terem que se preocupar com a sintaxe (Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015).

O Scratch pode ser usado como uma introdução à programação, especialmente entre os alunos mais jovens, mas também como uma forma de facilitar a transição para outros ambientes de programação, introduzindo e fomentando o interesse pela ciência da computação (Wolz, Maloney, & Pulimood, 2008). De acordo com uma pesquisa realizada por Meerbaum-Salant et al. (2013), a maioria dos alunos pode atingir um nível razoável de conceitos de Ciência da Computação através do Scratch. No entanto, surgem dificuldades ao ensinar tópicos específicos que requerem um maior nível de abstração.Isso pode, no entanto, ser resolvido se os alunos forem acompanhados no processo pelo professor.

O site do Scratch oferece suporte a uma rede mundial de usuários, tendo uma comunidade de programadores que permite aos usuários compartilhar projetos (Meerbaum-Salant, Armoni, & Ben-Ari, 2013). Scratch está disponível em mais de 50 idiomas, incluindo búlgaro, croata, inglês, grego, italiano, português, esloveno e turco. Também possui um editor offline

que permite que o programa seja descarregado para um computador e executado sem ter conexão à internet.

Snap!

O Snap! (https://snap.berkeley.edu/) também é uma linguagem de programação baseada em blocos visuais cujo design é baseado em Scratch, mas com alguns recursos adicionais. O Snap !, à semelhança do Scratch, é baseado na web, mas também tem a possibilidade de ser executado offline através de um navegador. Além dos recursos do Scratch, o Snap!adiciona listas de classe, procedimentos de classe, sprites de classe, trajes de classe, sons de classe e continuações de classe, tornando-o mais adequado para públicos mais avançados do que Scratch. Snap! está disponível em mais de 40 idiomas, incluindo búlgaro, croata, inglês, grego, italiano, português, esloveno e turco.

Alice

*Fig. 4. SEQ Picture * ARABIC 3* Alice 3 Code Editor

Alice https://www.alice.org/ é um ambiente de programação baseado em blocos que visa motivar os alunos a programar, estimulando sua criatividade por meio da criação de animações, narrativas interativas ou jogos 3D simples (Kelleher, Pausch, & Kiesler, 2007) O jogo permite a criação de mundos virtuais com um Editor de mundos virtuais, onde os alunos podem adicionar objetos 3D e adicionar funções e métodos aos objetos 3D existentes. Uma vez que o mundo virtual é criado, o aluno pode escrever o código para desenvolver a lógica do jogo / narrativa / animação (Sykes, 2007). Alice é uma boa linguagem de programação para alunos sem experiência anterior, pois permite que vejam os programas animados rodando enquanto escrevem seu código (Cooper, Dann, & Pausch, 2000).

Além disso, outro estudo que se concentrou em Storytelling Alice (um ambiente de programação baseado em Alice 2 com ferramentas criativas de escrita de histórias adicionais) descobriu que, embora Alice Genérica e Alice contadora de histórias fossem igualmente bem-sucedidas no ensino de conceitos de programação para raparigas, com Alice as raparigas contadoras de histórias estavam mais motivadas e aumentou seu tempo de programação, o que teve um impacto positivo na programação. Alice 3, a versão mais recente da série, está disponível em 13 idiomas, incluindo inglês, português, grego, esloveno e búlgaro, inglês, espanhol, português e alemão.

Tynker

Tynker https://www.tynker.com/ é uma plataforma de programação educacional baseada numa linguagem de programação visual de arrastar e soltar. Ao contrário de outros ambientes de programação apresentados, este é um produto comercial.Um aspecto que adiciona ao Scratch consiste em como ele aborda a codificação como um jogo, no qual os usuários são obrigados a criar um programa para fazer os personagens se moverem, interagirem e realizarem diferentes tarefas.Apresenta problemas que os jogadores precisam resolver para que as crianças comecem a reconhecer padrões (Geist, 2016). O Tynker também fornece um tutor integrado para dar instruções passo a passo, para que o aluno possa aprender como aplicar os conceitos de codificação. O aluno também é capaz de visualizar os blocos em código JavaScript, permitindo-lhes compreender o bloco em uma linguagem de programação baseada em texto. A Tynker oferece mais de 23 cursos de programação, 11 cursos para iPad e mais de 2.000 atividades de codificação, além de planos individuais e planos familiares (para até 4 membros). Está disponível apenas em inglês.

Fig.4. 4 Tynker User Interface

4.2.1. Aprendizagem de programação num ambiente baseado em jogos

A seguir, apresentamos algumas plataformas baseadas em jogos destinadas a desenvolver habilidades de codificação.

Code Combat

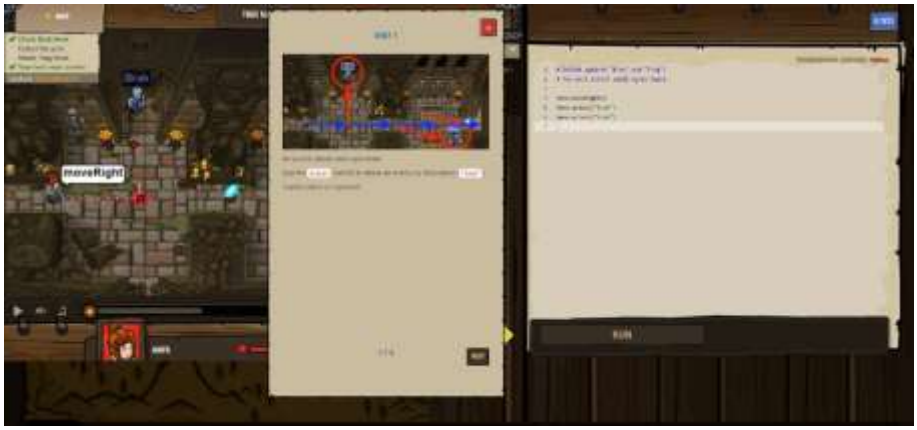


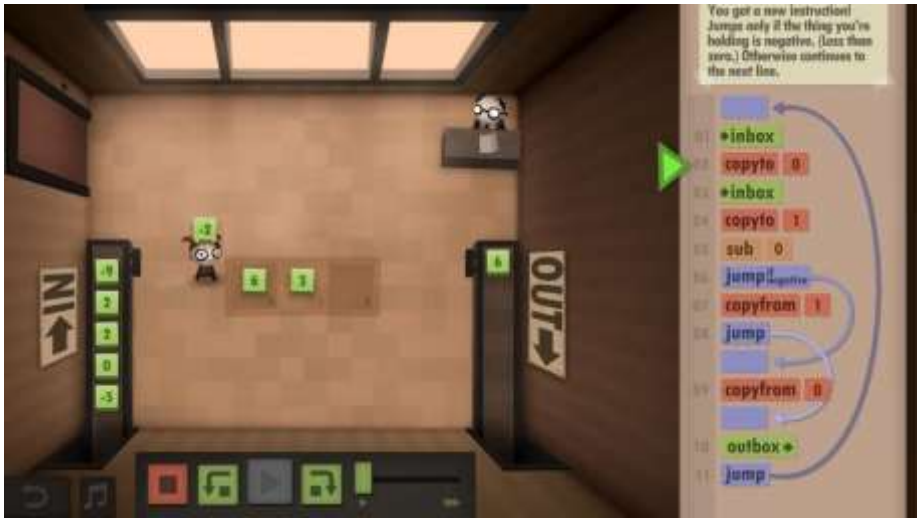
*Fig. 4. SEQ Picture * ARABIC 5* Code Combat Gameplay

Code Combat https://codecombat.com/ é um jogo de aventura baseado na web. Tem planos para alunos individuais e turmas, fornecendo também recursos para os professores usarem durante as aulas. Possui mais de 100 níveis gratuitos e somente para assinantes. Está disponível em mais de 50 idiomas, incluindo búlgaro, croata, inglês, grego, italiano, português, esloveno e turco. Os alunos devem escrever seu próprio código (em JavaScript ou Python) para fazer os personagens se moverem, interagirem e realizarem diferentes tarefas. Não é baseado em blocos, exigindo que os alunos escrevam seu próprio código. Ele fornece dicas ao longo do caminho e apresenta conceitos gradualmente. O jogo está dividido em 5 mundos diferentes, introduzindo diferentes conceitos conforme o jogador avança no jogo: 1) Kithgard Dungeon - sintaxe, métodos, parâmetros, strings, loops e variáveis; 2) Floresta do Sertão - If / else, lógica booleana, operadores relacionais, funções, propriedades do objeto, tratamento de eventos, tratamento de entradas; 3) Deserto de Sarven - aritmética, contadores, loops enquanto, quebrar, continuar, matrizes, comparação de strings, encontrar min / max; 4) Cloudrip Mountain - literais de objeto, método remoto, invocação, for-loops, funções complexas, desenho, módulo; 5) Geleira Kelvintaph - Técnicas Avançadas.

De acordo com Miljanovic et al. (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018) o conteúdo educacional do jogo compreende diferentes aspectos conceituais de design de algoritmos e resolução de problemas, conceitos fundamentais de programação (como Sintaxe e Semântica, Variáveis e Tipos de Dados Primitivos; Expressões e Atribuições, Entrada e saída, condicionais e iterativas, funções e parâmetros e recursão) e estruturas de dados fundamentais (como matrizes, agregados heterogêneos e tipos de dados abstratos).

Human Resource Machine

Human Resource Machine http://tomorrowcorporation.com/humanresourcemachine é um jogo de quebra-cabeça baseado numa linguagem de programação visual. Neste jogo, o jogador tem que automatizar uma tarefa programando um funcionário de escritório,



progredindo por quebra-cabeças cada vez mais difíceis (Johnson & all, 2016). O jogador deve criar uma sequência de movimentos, arrastando e soltando instruções, com novos comandos sendo introduzidos gradualmente para realizar operações mais complexas. Por meio dessa linguagem de programação visual, ele ensina conceitos fundamentais de programação por meio da comparação de algoritmos.

LightBot

LightBot https://lightbot.com/ é um jogo de quebra-cabeça com uma mecânica semelhante à máquina de recursos humanos, exigindo raciocínio lógico para avançar de nível. O jogador deve guiar um robô para iluminar peças e resolver 40 níveis. Os comandos usados no Lightbot aparecem como ícones.

Embora não haja aprendizagem explícita de uma linguagem de programação, os jogadores "... desenvolvem uma compreensão de sequenciamento e implementação de algoritmos" (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018) (p.6) por meio de um foco na resolução de problemas. No entanto, através do jogo os alunos aprendem diferentes conceitos fundamentais de programação, como estratégias condicionais, iterativas, recursão e depuração. Como o espaço para os blocos é limitado, o aluno também deve considerar a eficiência do código.

Um estudo conduzido por Mathrani et al (Mathrani, Christian, & Ponder-Sutton, 2016) usou o Lightbot com alunos do ensino médio e descobriu que os alunos gostavam de jogar e que essa era uma maneira eficaz de aprender construções de programação como funções, procedimentos, condicionais e recursão. A maioria dos alunos envolvidos também concordou que a abordagem foi uma forma eficaz de eles entenderem conceitos que, de outra forma, seriam mais difíceis de entender.

A Viagem da May

A viagem da May é o único jogo desta lista que é especificamente dirigido a alunas do Ensino secundário. É um puzzle 3D no qual os jogadores percorrem um labirinto utilizando a linguagem comum de programação do jogo, inspirada pela Java. O jogo foi desenvolvido para atrair raparigas para a área da Computação, ao ensinar-lhes os princípios básicos de programação.

*Fig. 4. SEQ Picture * ARABIC 8 Jogo A Viagem da May*

O criador aplica pseudocódigo para facilitar a transição entre linguagem de programação visual e de programação real. O conteúdo de aprendizagem inclui instruções básicas, sequências lógicas, loops, variáveis, statements if, comparadores e lógica Booleana, operações com números inteiros e operações em strings (Jemmali, 2016).

Existem duas fases no jogo, uma fase de exploração com as mecânicas de um jogo típico, e uma fase de programação. Na fase de programação, a interface está dividida, de forma a que o jogador consiga escrever o código e, ao mesmo tempo, ter feedback visual da execução do programa. Durante a fase de exploração são também fornecidas dicas para o código. Na história, a heroína é uma rapariga que vive num mundo que se está a desmoronar, está afastada da sua melhor amiga, e precisa de compor o mundo do jogo e resolver mistérios. As várias partes do mistério vão sendo reveladas gradualmente, motivando os jogadores a continuarem. Uma vez que as adolescentes são o principal grupo-alvo, a personagem principal do jogo assemelha-se a uma rapariga do ensino secundário, de forma a que o jogador se possa projetar nela. O design também procurou ir ao encontro da preferência da maioria das pessoas do género feminino (plataforma com mais luz, esquema de cores mais quentes, ilustrações menos agressivas). O grupo que testou o jogo pareceu também gostar da narrativa.

Snack Bar sem insetos

O Snack Bar sem insetos é um jogo sério de investigação com uma abordagem de agarrar e largar blocos, com foco na resolução de problemas. Os resultados de aprendizagem estão relacionados com a manipulação de variáveis, sequência de ações, condicionais e iterativas e estratégias de depuração (Vahldick, 2017).

*Fig. 4. SEQ Picture * ARABIC 9* Jogo “Snack Bar sem insetos”

No jogo, a personagem principal trabalha num snack-bar e deve cumprir várias missões usando código. Uma inovação integrada neste jogo é uma ferramenta para os professores monitorizarem o progresso dos alunos. Há também uma abordagem de gamificação, segundo a qual o aluno ganha pontos à medida que avança nos jogos e encontra formas de melhorar as suas soluções para os problemas apresentados. O jogo incorpora um assistente de dicas e um sistema administrativo para que o professor veja quem são os alunos que precisam de ajuda e, consequentemente, enviarem dicas personalizadas. A presença de um professor é, portanto, considerada fundamental.

Robot ON!

Robot ON! É um jogo-puzzle dirigido a estudantes do Ensino superior que estão a aprender C++. Neste jogo, o jogador é um cientista que deve ativar um 'Mech Suit' através de uma série de tarefas. Assim que o jogador termina um nível, ele ativa um novo sistema de robot. Os professores podem criar seus próprios níveis usando outras linguagens de programação. O jogador tem diferentes ferramentas, que são codificadas por cores, sendo que a cor do código corresponde a diferentes ferramentas. O jogador é apresentado a diferentes ferramentas, uma de cada vez, acompanhadas por tutoriais (Miljanovic & Bradbury, 2016).

Em vez de se focar na escrita de código, este jogo é focado na compreensão da

*Fig. 4. SEQ Picture * ARABIC 10 Jogo* Robot ON!

programação, a fim de ensinar competências de depuração e possibilitar aos estudantes que entendam códigos escrito por outros. Conforme o jogo avança, o jogador recebe as seguintes ferramentas: 1) ferramenta ativadora (para aprender o fluxo de controlo); 2) ferramentas de comentador e não-comentador (para aprender o comportamento do código); 3) ferramenta de nomeação (para aprender o propósito variável); e 4) ferramenta de verificação (para aprender o fluxo de dados).

Jogo educacional do Pacman

O Jogo Educacional do Pacman é um jogo sério de pesquisa projetado para ensinar algoritmos de pesquisa, permitindo que os alunos vejam como diferentes algoritmos de pesquisa se comportam, através de uma representação gráfica anotada (Grivokostopoulou, Perikos, & Hatzilygeroudis, 2016).

O jogo tem duas modalidades:

1) Modalidade Educacional: o estudante lê uma descrição textual e ver o fluxograma gráfico e o pseudocódigo de um algoritmo à sua escolha. O jogo também permite que o estudante aprenda o algoritmo através de visualizações de vários Labirintos do Pacman.
2) Modo de jogo: o estudante tem de resolver vários níveis de labirintos, lidando com as suas várias condições e respeitando um limite de tempo. Estes níveis foram construídos de forma a que o estudante tenha de aplicar um algoritmo de

*Fig. 4. SEQ Picture * ARABIC 11* Jogo educacional do Pacman

pesquisa específico para mover o Pacman pelo labirinto – é pedido ao jogador que, a partir de uma certa posição, chegue a um determinado local, movendo-o usando um algoritmo específico.

CMX

O CMX é um jogo MMORPG (Massive Multiplayer Online Role Playing Game) para aprendizagem de programação. No jogo, há duas equipas - crackers e hackers - que competem entre si para encontrar palavras-passe numa fábrica de lixo tóxico. As personagens são ajudadas por Senseis, que os apoiam na aprendizagem da linguagem de programação C. O jogo inclui três níveis de Senseis; o jogador acede ao nível seguinte depois de desbloquear uma senha no nível anterior. O jogo poderá contribuir para aumentar o conhecimento dos alunos sobre linguagem C – eles podem, não apenas responder a questões teóricas, mas também criar programas executáveis arrastando e soltando ferramentas, assim como escrever programas em C (Malliarakis, Satratzemi, & Xinogalos, 2014).

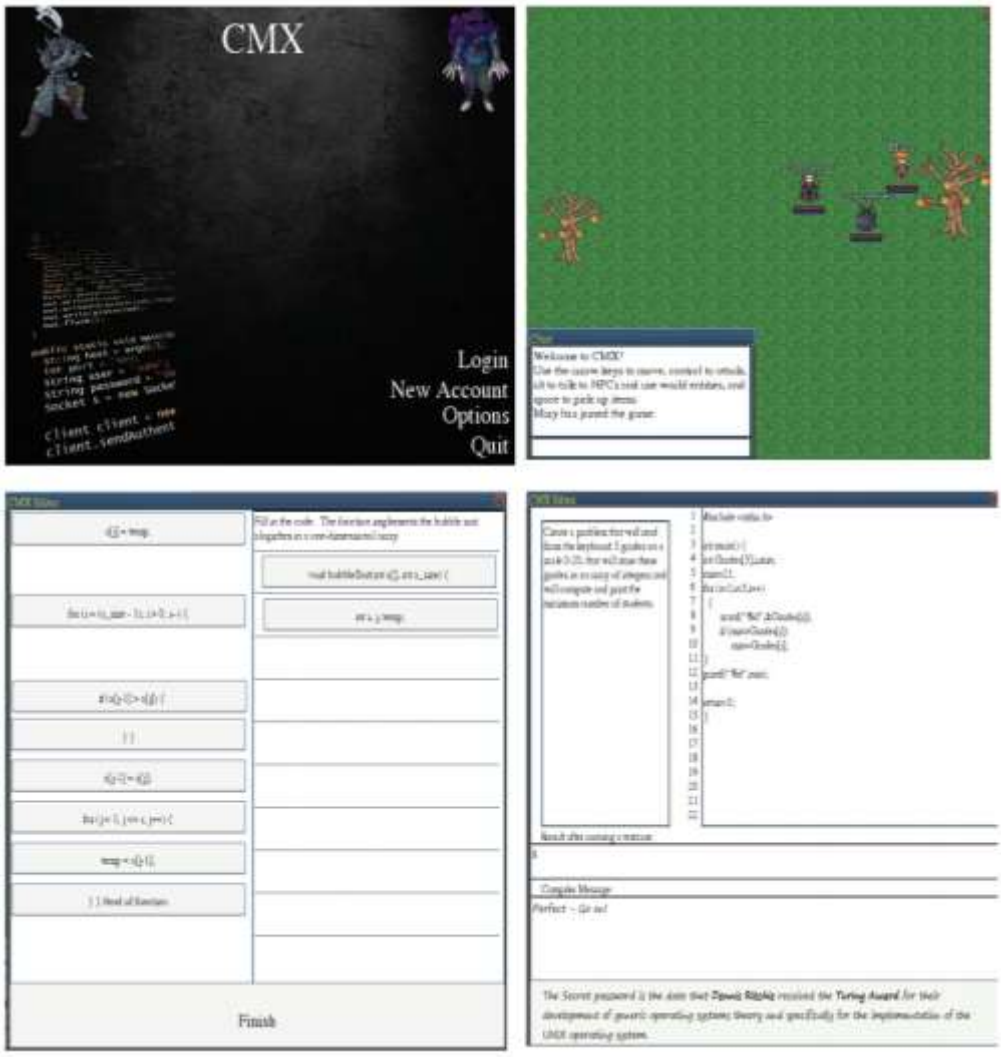


*Fig. 4. SEQ Picture * ARABIC 12 Jogo* CMX

JOGO	PLATAFORMA	GRUPO ALVO	LINGUAGEM	UNIDADES DE APRENDIZAGEM	TIPO DE JOGO
Combate de Código	Online	9+	JavaScript, Python	sintaxe, métodos, parâmetros, strings, loops e variáveis, If / else, lógica booleana, operadores relacionais, funções, propriedades de objeto, tratamento de evento, tratamento de entrada, aritmética, contadores, while-loops, break, continue, arrays, comparação de string, localizando min / max, literais de objeto, método remoto, invocação, for-loops, funções complexas, desenho, módulo	Jogo comercial (*Freemium*)
Máquina de Recursos Humanos	Windows, Mac OS X, Linux, iOS, Android	9+	Linguagem baseada em programação visual	Entrada e saída e condicionais e iterativos comparação de algoritmo	Jogo comercial (*Pago*) Puzzle
Lightbot	iOS, Android, Windows,	9+	Linguagem baseada em programaçã	Condicionais e iterativos e recursão Estratégias de	Jogo commercial (Pago)

	Mac OS X		o visual	depuração	Puzzle
A viagem da May	Windows	12-18	Linguagem de programação personalizada (inspirada por uma linguagem orientada para objetos, como Java)	Instruções básicas e lógica de sequência, loops, variáveis, if statements, comparadores e lógica booleana, operações em inteiros, operações em strings	Jogo de pesquisa Puzzle
Snack Bar sem insetos	Online	18+	Linguagem baseada em programação visual	Manipulação de variáveis, sequência de ações, condicionais e iterativas, estratégias de depuração	Jogo de pesquisa
Robot ON!	Windows	18+	C++	Sintaxe e semântica, tipos de dados variáveis e primitivos, expressões e atribuições, condicionais e iterativos, compreensão do programa	Gratuito/Jogo de pesquisa

Jogo educacional do Pacman	Windows	18+	Jogo para aprender algoritmo de pesquisa de IA	Algoritmos (descrição textual básica de algoritmos e fluxograma gráfico correspondente junto com o pseudocódigo)	Jogo de pesquisa
CMX	Windows	18+	C	Teoria sobre matrizes, saídas de variáveis após a execução de um programa de matriz, sintaxe de programas de matriz, entrada de valores de variáveis em uma matriz, cálculo da soma da matriz, cálculo do valor máximo da matriz,	Jogo de pesquisa MMORPG

Tabela 4.2. Comparação do ambiente dos vários jogos para ensino de programação

5. PLATAFORMA DO CODING4GIRLS PARA PROFESSORES E ALUNOS

5.1. Plataforma do Coding4Girls e abordagem de design-thinking

A combinação perfeita entre as TIC e os jogos sérios é apresentada no resultado O2 do projeto, *Promover o Desenvolvimento de Competências de Programação de Raparigas através do Jogos Sérios*, que se destina a desenvolver essas competências em jovens, especialmente do género feminino, do ensino básico e secundário, recorrendo a um software desenvolvido no âmbito do projeto.

Este software foi desenvolvido com o objetivo de superar a discrepância entre a participação de pessoas do género masculino e feminino na educação e profissões relacionadas com computação, recorrendo a uma metodologia que procura tornar a programação atrativa para todos. O software e os cenários de aprendizagem foram desenvolvidos para estimular o envolvimento de raparigas nas áreas da computação. Foram desenvolvidos seguindo uma abordagem de design-thinking; ou seja, seguindo um processo criativo que ajuda os estudantes a chegarem a soluções relevantes em colaboração com os seus colegas.

O processo de design é muito eficaz devido ao seu quadro estruturado para identificar desafios, recolher informação, gerar possíveis soluções, refinar ideias e testar soluções.
O processo é recursivo por natureza e exige interação. Cada etapa do processo exige uma revisão e uma evocação ao longo de uma experiência de aprendizagem para estimular a experimentação, a viabilidade da solução e a reflexão. (Luka, 2014)
Atividades com base num processo de design-thinking favorecem uma experiência altamente colaborativa, e fora dos limites da sala de aula. Os estudantes estão diretamente envolvidos na recolha de informação, criação de conhecimento, comunicação e apresentação.
No projeto Coding4Girls, a abordagem do design-thinking explora as principais vantagens da aprendizagem baseada em jogos para encorajar e motivar os alunos nos seus processos de aprendizagem. Recorre a alguns elementos de jogo importantes (por exemplo, sistema de pontos e desafios) e a cenários que tornam uma determinada atividade mais divertida, aumentando o grau de envolvimento e motivação dos estudantes e melhorando os resultados alcançados.
Em particular, o facto de permitir que os estudantes vejam os desafios enquanto os ajuda a ter uma visão geral do problema, antes de trabalharem numa solução detalhada, ajuda-os a pensar de forma empreendedora sobre as tecnologias digitais e a refletir sobre como elas podem ser usadas para resolver problemas do mundo real.
O software, desenvolvido no âmbito do projeto, consiste em duas partes diferentes interligadas: a Plataforma do Professor e o Ambiente de Jogo do Aluno.

5.2 A Plataforma do Professor

A Plataforma do Professor é uma plataforma web na qual os professores podem projetar e preparar cursos específicos para desenvolver as competências de programação dos estudantes, utilizando o Snap!.

Dentro da plataforma, cada professor tem à sua disposição uma área privada e pública. Na primeiro, eles podem preparar os seus próprios cursos, com base nas necessidades de seus alunos. A segunda área funciona como um repositório de todos os cursos criados por outros professores. O objetivo é compartilhar cursos, usar outros já feitos ou inspirar-se nas atividades de aprendizagem já preparadas por outros colegas (Fig. 5. 1). Todos os professores podem, no entanto, personalizar os cursos públicos de acordo com a especificidade das suas turmas e aulas.

Fig. 5.1 Plataforma do Professor a alguns dos cursos de programação disponíveis

Estes cursos constituem um grupo de atividades relacionadas tematicamente, todas ligadas a um tópico abrangente. O problema é apresentado logo no início do curso para os alunos possam fazer um brainstorming para chegar a soluções experimentais de forma colaborativa. Esta é uma fase muito importante do processo de aprendizagem, onde os alunos podem gerar novas ideias e ser imersos criativamente no processo de resolução de problemas.

A tela de brainstorming pode ser preparada por professores na Plataforma do Professor, usando algumas perguntas-chave ou alguns comentários e reflexões a serem usados durante a discussão dos alunos. O software oferece a possibilidade de criar post-its num quadro virtual, usando texto, imagens ou vídeo conforme se pode observar na Figura 5.2:

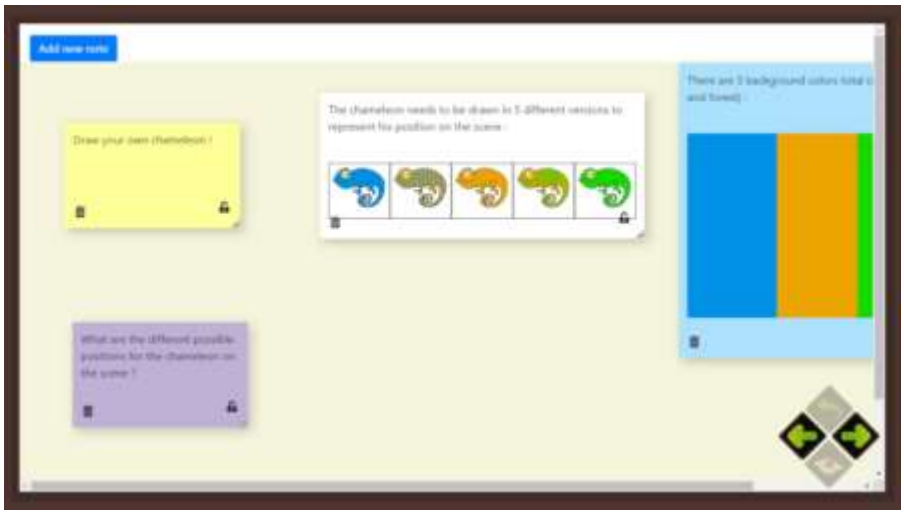


Fig. 5.2. Zona de brainstorming no Ambiente de Jogo do Aluno, na qual os estudantes podem deixar a sua contribuição e debater uns com os outros. .

Todos os post-its possuem um ícone de cadeado que pode ser aberto quando o post-it for editável ou bloqueado quando o post-it não puder ser editado. Apenas os professores têm a possibilidade de bloquear e desbloquear post-its; os alunos apenas sabem se uma publicação está aberta ou não, não podendo alterar a configuração desta.

Após a fase de brainstorming, os alunos ficam a conhecer as atividades, passo a passo; atividades específicas (apresentadas em ordem consecutiva) destinadas a apresentar as ferramentas necessárias para resolver o problema geral. As atividades de programação são apresentadas usando o quadro do Snap! que exibirá tarefas incompletas para desafiar os alunos a procurar a solução do problema e a apresentar a solução final.

Na Figura 5.3, é visível a estrutura de cada curso C4G publicado na Plataforma do Professor:

Fig. 5.3 Estrutura de um curso do C4G

As atividades do curso são chamadas de desafios, e são apresentadas ao estudante numa determinada ordem, definida previamente pelo professor (Figura 5.4). Por exemplo, se um professor quiser criar um curso de conhecimentos básicos de programação, a primeira atividade pode ser sobre o conceito de booleanos, a segunda sobre estrutura condicional e a última sobre loops. Tal permite que os professores desenvolvam caminhos de aprendizagem personalizados para os seus alunos na Plataforma do Professor e que os alunos sejam conduzidos gradualmente até ao objetivo final de aprendizagem no Ambiente de Jogo dos Alunos.

Os alunos precisam de, claro, jogar e resolver todos os desafios na ordem estabelecida pelos professores antes de chegar à solução final.

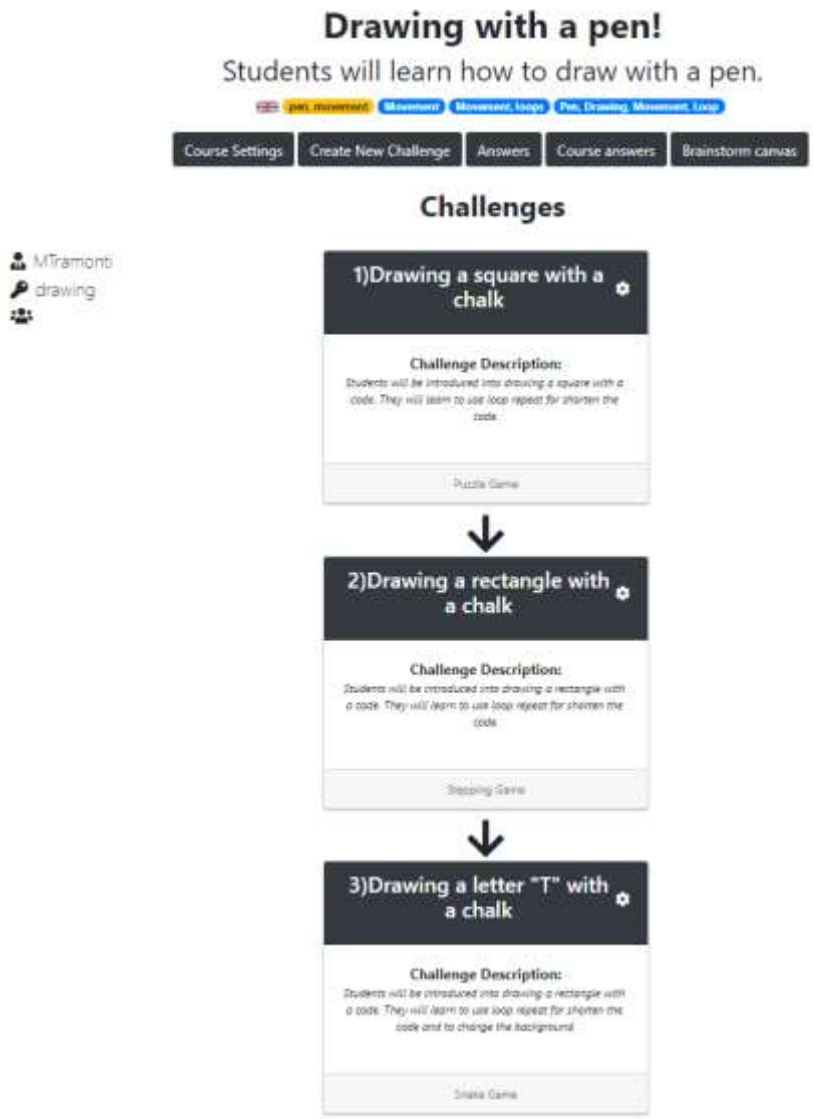


Fig. 5.4 Exemplo de uma série de desafios no curso de programação na Plataforma do Professor

Cada desafio pode estar associado a um minijogo desenvolvido pelo consórcio do projeto. O objetivo é ajudar os estudantes a compreenderem qual seria o resultado final de uma certa propriedade de código específica.

Por exemplo, se um desafio estiver ligado ao estudo da propriedade dos “loops”, o Match-3 é um dos minijogos desenvolvidos que podem ser associados a esse tópico.

Assim sendo, cada desafio pode ter associado, para além do quadro do Snap! para resolver a tarefa, um minijogo que o estudante precisará de jogar seguindo uma página de instruções

definidas pelo professor, aquando da sua preparação, na Plataforma do Professor (Figura 5.5).

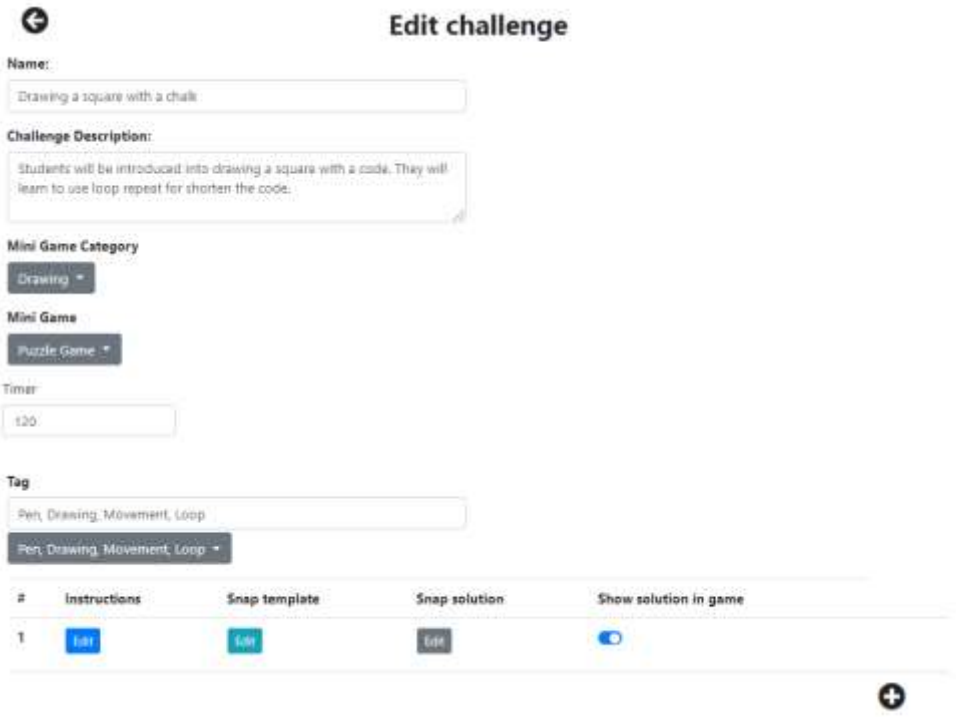


Fig. 5.5 Visão interna de um desafio na Plataforma do Professor

Todos os jogos oferecidos (atualmente 11) aos alunos estão relacionados e simplificam os conceitos de programação reais sobre os quais os cursos e cenários C4G foram projetados. No entanto, não é obrigatório incluir um minijogo no desafio. Tal trata-se de uma escolha do professor.

Cada desafio é constituído por três partes:

1. Instruções, onde o professor pode dar pistas ou explicações sobre a tarefa a realizar pelos seus alunos.

2. Template do Snap! onde os alunos tentarão resolver a tarefa atribuída usando a programação em bloco do Snap! no Ambiente de Jogo dos alunos

3. Solução, sendo que os professores podem decidir mostrar ou não a solução final da tarefa.

Se o desafio for complexo, em termos de conhecimento de programação, pode ser estruturado em mais de uma tarefa (Figura 5.6). Dessa forma, os professores poderão

conduzir, passo a passo, os seus alunos, por diferentes fases, até estes resolverem uma atividade complexa que se divide em partes menores.

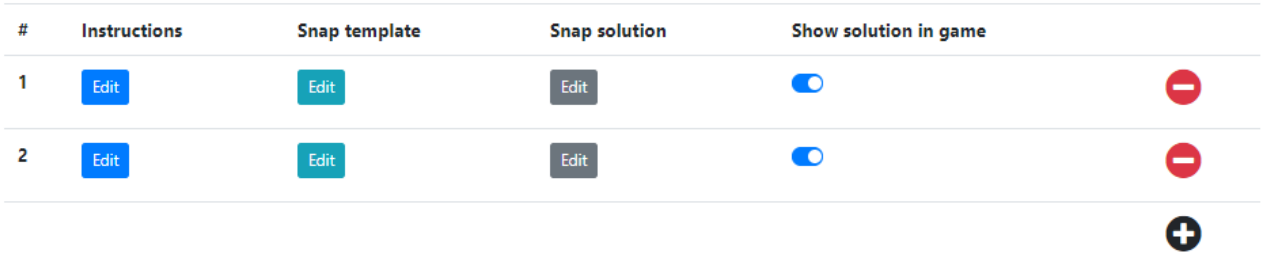


Fig. 5.6 Divisão de um desafio complexo em tarefas mais pequenas

Além disso, os professores podem reunir todas as respostas dos alunos no Snap! após eles as submeterem em cada um dos desafios, usando para tal uma tabela na Plataforma do Professor. A tabela contém a lista dos utilizadores e os respetivos detalhes para cada desafio existente no curso. Cada linha contém o nome de utilizador, nome, apelido, nome do desafio ou desafios resolvidos.

Se o estudante não tiver enviado uma solução, a coluna "Link da solução" ficará vazia. Caso contrário, a linha será destacada e a palavra “Solução” aparecerá na última coluna e, ao clicar nela, o Professor poderá aceder à submissão do estudante (Figura 5.7).

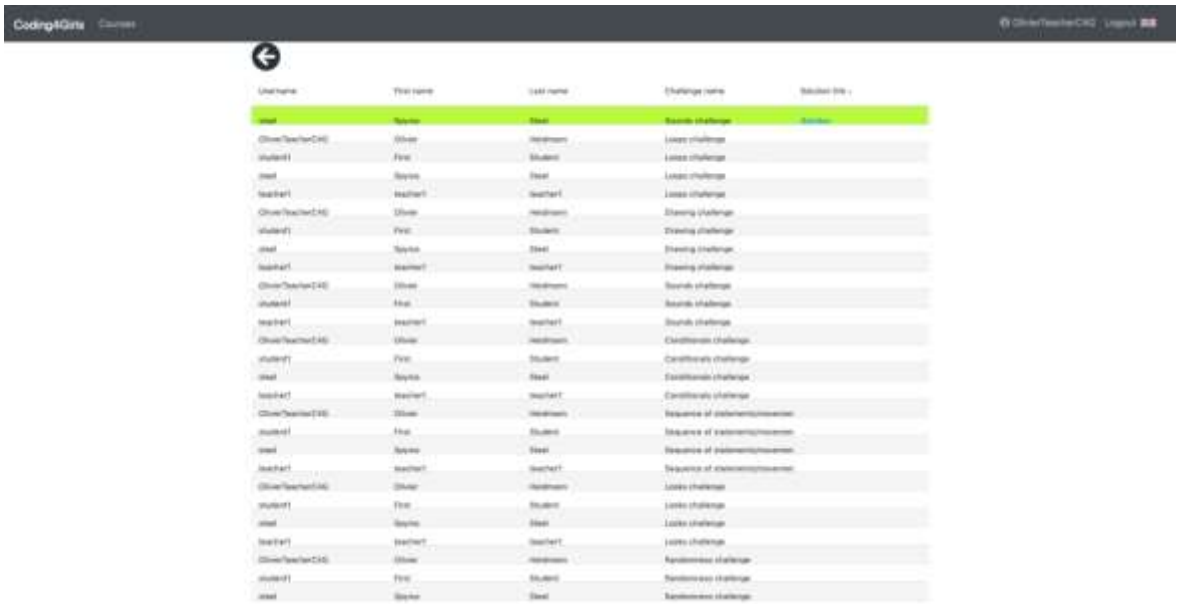


Fig. 5.7 Respostas com solução aos desafios, submetidas pelos estudantes

Desta forma, os professores conseguem monitorizar e acompanhar o progresso e resultados dos seus estudantes.

5.3. Ambiente de Jogo do Aluno

Os alunos terão acesso aos desafios através do Ambiente de Jogo do Aluno, que é um software de videojogos 3D Unity para download, no qual os alunos podem descobrir e concluir os cursos preparados pelos seus professores de maneira divertida, envolvente e lúdica.

Os cursos, utilizando elementos da abordagem de design thinking, apresentam aos alunos uma questão abrangente para resolver e apresentam as ferramentas e as instruções (passo a passo) para que a resolvam. Os desafios, conforme mencionado acima, são elaborados pelos professores e publicados na Plataforma do Professor, mas os alunos têm acesso a eles através do Ambiente de Jogo do Aluno.

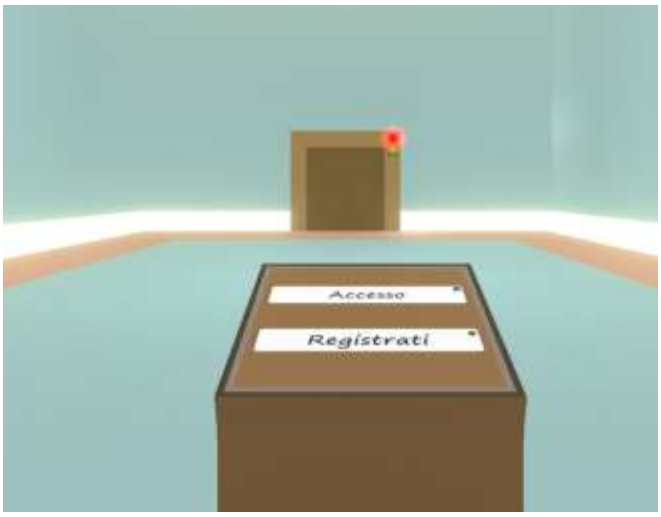


Fig. 5.8 A primeira sala para aceder ao Ambiente de Jogo do Aluno

Os alunos podem ingressar num determinado curso introduzindo o código correto no respetivo terminal. O código é único e deve ser fornecido por professores que já prepararam o curso na Plataforma de Professores. O curso, uma vez submetido, fica visível no topo do portal, que indica também o curso atualmente selecionado (Figura 5.9).

Fig. 5.9 Segunda sala, com dois terminais: um para o estudante se juntar, pela primeira vez, ao curso, e outro para aceder ao curso no qual já está registado

Os alunos são incentivados a projetar e codificar jogos que atendam a necessidades ou problemas específicos (dependendo da escolha dos professores). É uma “abordagem de entrada baixa com teto alto” (“low entry high ceiling approach”), que permite aos alunos

começar com problemas fáceis e irem progredindo até se envolverem em tarefas mais desafiadoras. Os professores projetam e apresentam aos alunos cenários "incompletos" nos quais uma solução está parcialmente pronta, desafiando-os a concluir a tarefa. Portanto, os professores podem preparar módulos pequenos e adaptáveis, usando o software Coding4Girls (C4G) adequado às necessidades e conhecimentos de seus alunos.

Seguindo a abordagem do Design Thinking, a equipa do projeto preparou alguns cursos e cenários de aprendizagem projetados para desafiar os alunos a resolverem um problema específico de programação. Atualmente, os 21 cenários de aprendizagem, reunidos no relatório "Coletânea de fichas de aprendizagem através de game-design dirigida a professores" foram readaptados à abordagem de design thinking seguindo a estrutura do software C4G.

Os cenários têm diferentes níveis - do básico ao avançado - e estão disponíveis em inglês, esloveno, italiano, croata, búlgaro e grego.

As tarefas podem ser resolvidas de forma individual e colaborativa estimulando a discussão em grupo na sala de aula ou, virtualmente, no Ambiente de Jogo dos Alunos. O software oferece um espaço de brainstorming onde os alunos podem discutir e compartilhar ideias, publicando e gerindo post-its com conteúdo multimédia, como mostrado na Figura 5.11. Todos os alunos envolvidos no curso podem escrever a sua contribuição, sendo que esta é visível para todos os inscritos no curso em questão.

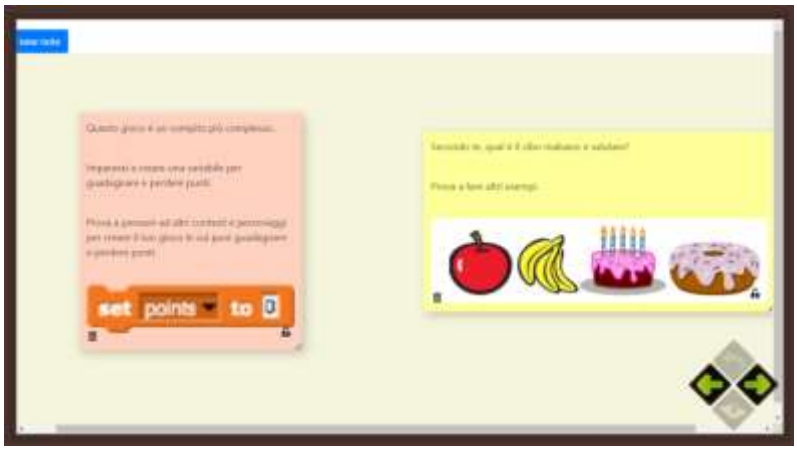


Fig. 5.11: Área de brainstorming no cenário de aprendizagem *Comprar comida para um piquenique* no Ambiente de Jogo do Aluno

A figura que se segue mostra o painel de desafios no Ambiente de Jogo do Aluno do curso criado pelo professor na Plataforma do Professor. Os números de 0 a 13 representam os desafios. Neste painel, o desafio número 0 representa as instruções gerais fornecidas no início do jogo e abri-lo levará os alunos às instruções do problema global, e ao template Snap! relacionado, seguido pela tela de brainstorming. Os desafios restantes, do número 1 ao 13, representam as tarefas elaboradas pelos professores na sua plataforma.

Fig. 5.12. Painel de desafios no Ambiente de Jogo do Aluno

Ao selecionar um deles, os detalhes do desafio aparecerão na parte inferior da tela, com o nome do desafio à esquerda, o minijogo associado ao desafio no meio e um botão para iniciar o desafio. A Figura 5.12 mostra o nome do desafio, correspondendo ao tema a ser estudado, neste caso, “Desafio condicional”. Além do tópico de desafio, o sistema mostra o minijogo 3D associado a ele, por ex. "Encontra o teu caminho".

Os professores podem decidir qual minijogo (opcional) os alunos jogarão, selecionando um dos minijogos existentes, como Match3, Dados, Inventário, Encontra o teu caminho, jogo de pisar, Jogo do Som, Jogo da Cobra, Jogo do puzzle, jogo de correspondência de padrões e quizzes de escolha múltipla.

Fig. 5.13 Exemplo de um dos minijogos disponíveis - Dados

Cada desafio, no Ambiente de Jogo dos Alunos, é estruturado da seguinte forma: um minijogo introdutório ilustrando o conceito de programação; uma página de instruções para a tarefa a ser realizada no Snap; dois quadros do Snap! - um para ser usado para resolver a tarefa e outro para exibir a solução da atividade.

A página instrucional é em HTML (Figura 5.14), geralmente enriquecida por imagens ou vídeos, para apresentar o contexto e objetivos específicos da tarefa de aprendizagem a ser desenvolvida no Snap!

Fig. 5.14 Um exemplo da página instrucional do desafio, no Ambiente de Jogo do Aluno

A esta página segue-se um quadro do Snap! Figura 14), com base no template disponibilizado pelo professor, onde constará a atividade ou problema de programação a ser executada ou resolvida pelos estudantes.

Fig 3.15 Exemplo de um template do Snap! disponibilizado pelo professor para resolver a tarefa de programação no Ambiente de Jogo do Aluno

Outro quadro do Snap! irá ser disponibilizado para exibir a solução para o desafio proposto. No entanto, apenas o professor pode decidir se o quadro com a solução é visível ou não; tal irá depender do processo de ensino escolhido por ele.

Fig. 5.16 Exemplo de um quadro do Snap! com a solução do desafio proposta pelo professor para os estudantes, disponível no Ambiente e Jogo do Aluno

Uma vez que cada curso reúne mais desafios, essas etapas serão repetidas tantas vezes quanto o número total de desafios identificados pelos professores. Isso permite dividir uma atividade de programação complexa em etapas elementares simples, onde a resposta e / ou a solução para a atividade anterior se torna o template no qual a próxima atividade deve ser executada.

No final, o curso é constituído por um certo número de desafios que possibilitam o ensino da programação através de um processo de progressão incremental.

Depois de os alunos terem concluído todos os desafios do curso (Figura 7), eles são conduzidos de volta ao problema de programação inicial e ser-lhes-á pedido que resumam o novo conhecimento que acabaram de adquirir. No final do curso, os jogadores podem ver também todas as soluções para o problema proposto pelos outros alunos, o que pode suscitar momentos de debate.

A abordagem e as ferramentas propostas pelo C4G permitem que os alunos desenvolvam, melhorem e reforcem as competências de programação através de atividades de formação personalizadas e adequadas às suas necessidades que incluem tarefas pedagógicas e divertidas.

6. EXEMPLOS DE LIÇÕES (FICHAS DE APRENDIZAGEM)

6.1. Fichas dos cenários de aprendizagem

No âmbito do projeto Coding4Girls foram desenvolvidas 22 fichas de cenários de aprendizagem, uma para cada um deles. Estas descrevem as atividades de aprendizagem combinada, que incluem as abordagens de jogos sérios e design-thinking. As fichas de aprendizagem estão disponíveis em Inglês e nos idiomas oficiais dos países parceiros do projeto – Búlgaro, Croata, Grego, Italiano, Português, Esloveno e Turco. Todas elas estão disponíveis no website do projeto (https://www.coding4girls.eu/results_03.php)

As fichas de aprendizagem apresentam, de forma concisa, a informação necessária para que professores e instrutores integrem as metodologias de jogos sérios e design-thinking nas suas práticas de ensino. As fichas seguem a abordagem de aprendizagem com base em jogos do Coding4Girls e incluem informação sobre cada atividade desenvolvida para desenvolver as competências de programação de rapazes e raparigas. Cada ficha inclui a seguinte informação:

- Objetivos gerais de aprendizagem da atividade
- Conceitos abordados na atividade de aprendizagem
- Objetivos de aprendizagem específicos
- Resultados de aprendizagem esperados
- Guia passo-a-passo da abordagem de aprendizagem baseada em jogos
- Métodos de avaliação para verificar o conhecimento desenvolvido
- Questões para iniciar o debate entre alunos, numa lógica de colaboração

Os professores podem usar os cenários e jogos seguindo a ordem proposta ou podem selecioná-los livremente, de acordo com as suas preferências e necessidades. Além disso, os professores podem ter de adaptar certos cenários ao conhecimento dos seus alunos, de acordo com os conceitos de matemática aprendidos – por exemplo, sistema de coordenadas, coordenadas, números negativos, frações, etc.

As folhas de aprendizagem abordam tanto a funcionalidade genérica do jogo sério proposto, incluindo processos de interação do usuário e geração de feedback, bem como descrições de todas as atividades de aprendizagem que serão implementadas no jogo sério proposto.

As folhas de aprendizagem preparadas variam entre as mais básicas, com apenas um conceito de programação, e as mais avançadas, com vários conceitos de programação. A tabela a seguir apresenta a ordem de atividades proposta.

Tabela 6.1. Lista de fichas de aprendizagem desenvolvidas no projeto Coding4Girls

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

Todas as fichas de aprendizagem estão no Anexo 1. No Anexo 2, são disponibilizados códigos para todos os jogos apresentados na secção pública do ambiente Coding4Girls.

6.2. Exemplos do uso do ambiente de jogo desenvolvido no âmbito do projeto Coding4Girls

O Ambiente de Jogo do Aluno é um videojogo sério baseado em Unity disponível para Windows e Mac OS X. Ao jogar, o utilizador depara-se, em primeiro lugar, com uma cena introdutória que apresenta a isenção de responsabilidade europeia exigida e o logotipo Erasmus +; estes serão exibidos por 4 segundos.

Depois disso, o jogador é encaminhado para uma sala vazia chamado átrio, na qual controla um avatar invisível e vê o mundo do jogo através de uma perspetiva de primeira pessoa. Uma consola é colocada no meio da sala, permitindo que o utilizador faça login no servidor C4G usando suas credenciais ou criando uma nova conta. Depois de a sessão estar iniciada, abre-se uma porta e o jogador pode prosseguir para outra sala, na qual se pode inscrever num curso novo ou num que já tenha sido ativado. Assim que o curso desejado for selecionado, o

utilizador passará por um portal flutuante para prosseguir com o jogo e aceder ao conteúdo específico do curso selecionado.

Cada curso é formado por uma coleção de desafios de programação ligados uns aos outros, que devem ser resolvidos na plataforma do Snap! sendo cada um deles ilustrado por um minijogo. Idealmente, cada desafio deve ilustrar um aspeto ou passo da lição a ser aprendida no curso.

O C4G decidiu alguns dos conceitos básicos de programação a serem abordados, sendo que cada um deles é ilustrado por um minijogo: loops, condicionais, variáveis, statements, paralelismo, operadores, eventos. Além disso, os professores podem escolher substituir o minijogo por um quizz de escolha múltipla, ou então não incluir nada, deixando apenas os exercícios Snap! nos desafios.

Fig. 6.1 A categoria do minijogo, disponível na Plataforma do Professor

Cada desafio inclui tarefas de programação para serem resolvidas no quadro do Snap! sendo que estas podem ser associadas a minijogos exibidos no painel de desafios (Figura 6.2.).

Fig. 6.2 Painel dos desafios no Ambiente de Jogo do Aluno

A integração de um minijogo num desafio específico tem o propósito de ilustrar o conceito de programação por detrás da tarefa planeada pelos professores. Não é obrigatório incluí-lo. Tal significa que o professor pode decidir ou não incluir um minijogo para seus alunos no desafio e pode selecionar qual será o minijogo, selecionando-o de uma lista previamente desenvolvida.

Um minijogo deve ser associado ao desafio se se considerar que acrescenta valor, ajuda à compreensão, funciona como prova de conceito, ajuda à visualização da mecânica e, claro, se engajar e motivar os alunos para os seus percursos de aprendizagem.

No momento, existem 11 minijogos diferentes, que vão desde um jogo Match3 a um questionário de múltipla escolha.

Fig. 6.3. Minijogos disponíveis na Plataforma do Professor

Quando um utilizador entra num desafio associado a um minijogo, será reconduzido para a localização na qual este se desenrola. Os minijogos têm lugar nos locais mais variados: desde praias até a montanhas com neve.

Fig. 6.4 Exemplo de localização de um dos minijogos

De entre dos minijogos disponíveis, podem destacar-se o jogo Match3, o jogo dos dados, o jogo do inventário, o jogo Encontra o teu Caminho, o jogo de pisar, o jogo do som, o jogo do puzzle, o jogo de correspondência de padrões e um quizz de escolha múltipla.

Cada minijogo tem as suas próprias dificuldades, e o leque de jogos disponíveis procura ter opções adequadas às várias faixas etárias e aos vários tópicos de programação ensinados em diferentes cursos.

Um dos jogos mais simples é o da cobra, que se desenrola à beira-rio, e no qual o jogador tem de seguir o sinal de seta de madeira até um círculo vermelho para iniciar a atividade (Figura 6.5).

Fig. 6.5 Exemplo de localização de um dos minijogos (jogo da cobra)

O jogo da cobra (Figura 6.6) ocorre na água. O jogador controla a cobra usando as 4 setas do teclado e tenta sobreviver o maior tempo possível. A cobra aumenta de tamanho à medida que come os pontos verdes brilhantes, mas pode morrer se eles tocaram na sua cauda, se ela tocar na borda da tela ou comer um ponto vermelho. Cada ponto verde comido adiciona um ponto à pontuação.

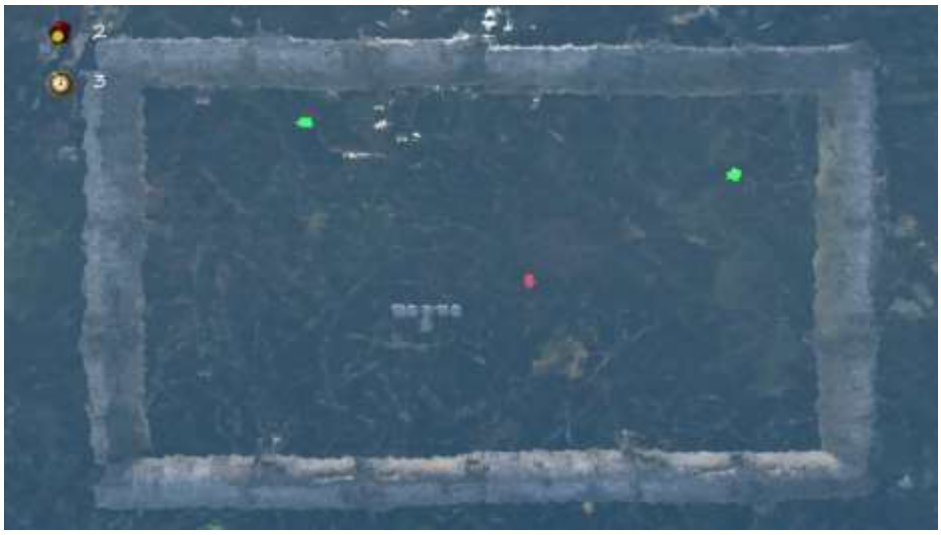


Fig. 6.6 Jogo da cobra

Este minijogo simples ilustra eficazmente conceitos de programação como loops, movimento, mudança de gráficos, etc.

Um outro exemplo é o jogo Match3. Este desenrola-se numa região elevada com neve, na qual os participantes podem vaguear livremente.

Fig. 6.7 Localização do jogo Match3

O minijogo Match3 é baseado numa atividade Match3 típica, na qual o jogador tem de arrastar e largar quadrados adjacentes de forma a fazê-los desaparecer se formarem um grupo de três azulejos (ou mais) iguais.

Fig. 6.8 Minijogo do Match3

O jogo dos dados passa-se na Floresta. A parte interativa do minijogo decorre dentro de uma cabana abandonada, na cadeira à frente da porta. Existe um tabuleiro de madeira com dois dados. O jogador e a inteligência artificial do jogo devem lançar os dados, e aquele que alcançar o valor mais alto, através da soma dos valores dos dois dados, ganha a ronda.

Fig. 6.9 Jogo dos dados

O jogo do inventário decorre no campo, perto de uma pequena floresta. Esferas coloridas flutuam por ali e o jogador deve apanhá-las, seguindo para tal as instruções deixadas pelo professor na tabuleta de madeira.

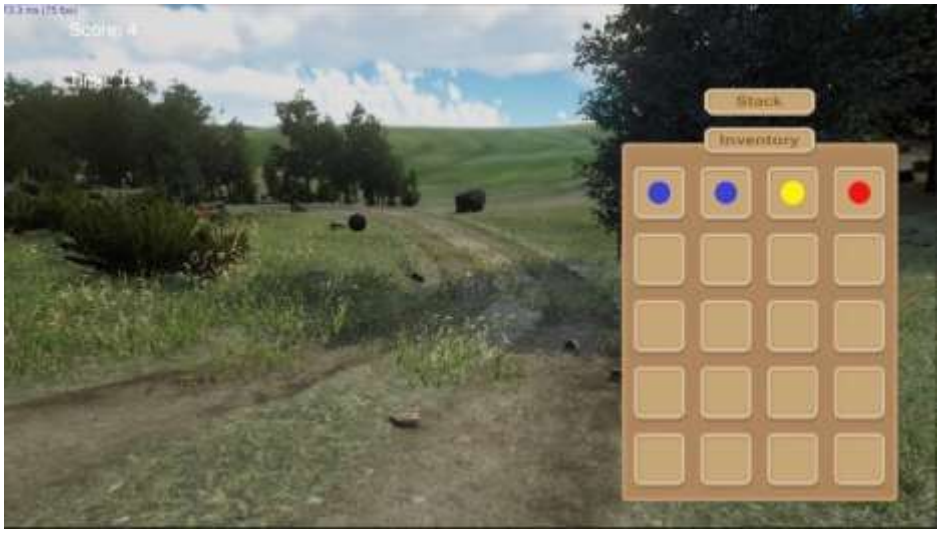


Fig. 6.10 Jogo do inventário

O jogo de pisar decorre ao pôr do sol, numa praia de uma ilha. O jogador tem de responder à questão escrita na rocha grande (que se encontra no centro da praia) pisando nas rochas com as letras para soletrar a resposta.

Quando o jogador pisa numa rocha com uma letra que faça parte da resposta, a letra será realçada a verde, e irá aparecer na outra rocha grande, ao lado daquela na qual a pergunta está escrita. Se o jogador pisar numa rocha errada, a letra será realçada a vermelho e ele terá de recomeçar a soletrar a resposta.

Fig. 6.11 Jogo de pisar

O quiz de escolha múltipla tem lugar perto de um penhasco e, no início, os jogadores podem vaguear livremente para iniciarem o jogo.

Surgirão questões no topo do penhasco, sendo que cada campo exibe uma resposta possível, juntamente com uma imagem (no caso de tal se justificar).

O ícone com a medalha representa a pontuação atual do estudante, com base nas respostas corretas, e o relógio vai contando o tempo até ao final do desafio.

A seta, localizada na parte inferior da tela, permite ao jogador prosseguir para a resposta seguinte.

Fig. 6.12 Quiz de escolha múltipla

O jogo do puzzle desenrola-se numa área rochosa do mapa. O objetivo é encontrar os painéis que se encontram dispersos pela área e completar o puzzle criando uma linha com início no círculo branco até ao quadrado vermelho do painel.

Fig. 6.13 Um painel no jogo do puzzle

O jogo de correspondência de padrões decorre à beira-rio. Existem algumas caixas com números escritos nelas que flutuam em direção ao mar. O objetivo é que o jogador as apanhe, para depois completar uma certa operação matemática.

Ou seja, os jogadores precisam de usar os números flutuantes e as operações disponibilizadas no jogo para alcançarem um número pré-definido, escrito no quadro azul.

O número alvo e os números flutuantes são gerados aleatoriamente, mas de forma a que seja sempre possível fazer combinações e chegar ao resultado pretendido.

Dependendo do professor, as operações disponíveis podem ser as seguintes:

- Operações básicas (+, -, *, /)
- Operações avançadas (poder, raiz quadrada, módulo)
- Trignometria (Cos, Sin, Tan)
- Operações booleanas (AND, OR, XOR).

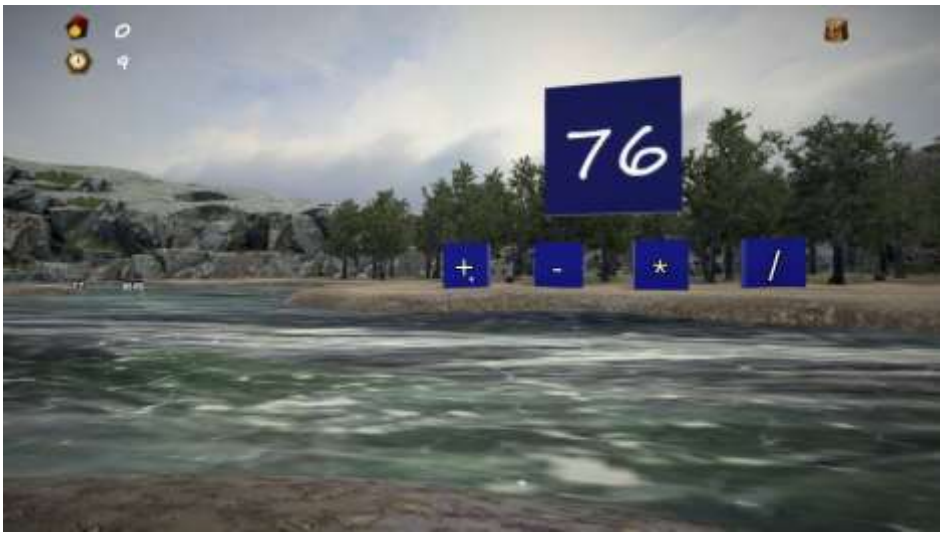


Fig.6.14 Jogo de correspondência de padrões

Outro exemplo de minijogo é o jogo dos sons, que requer que os jogadores prestem muita atenção de forma a serem bem-sucedidos nas suas tarefas. Desenrola-se numa floresta tropical. O jogador tem de encontrar os cinco painéis espalhados pelo espaço, e, para cada um deles, completar um puzzle com base nos sons da selva.

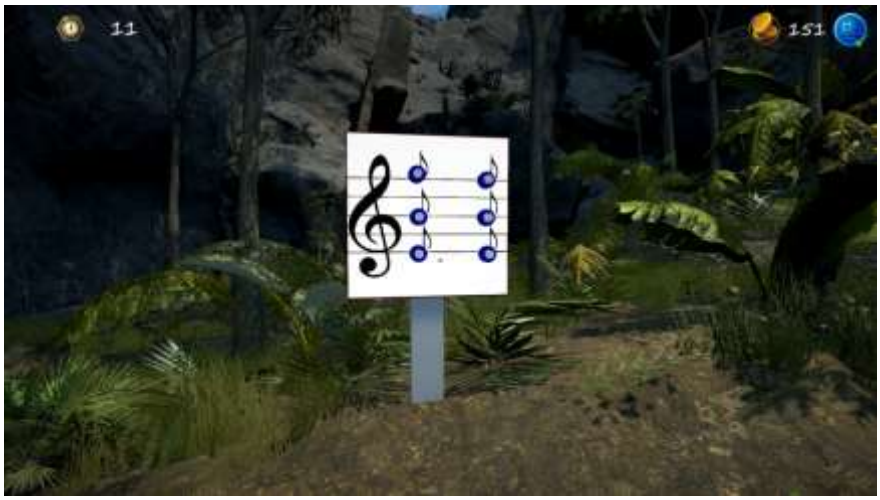


Fig. 6.15 Um dos painéis no jogo dos sons

Cada coluna do painel representa um som, e cada linha representa o tom do som. Os botões no topo são para os tons agudos e os da base são para os tons graves. Quando o jogador se encontra perto de um painel, uma combinação de três possíveis sons irá tocar (um canto de pássaro grave, um canto de pássaro médio e um canto agudo). A combinação de

cantos de pássaros é repetida em um loop, cada repetição separada pelas outras por um silêncio de 3 segundos.

O jogador terá de prestar atenção à ordem na qual os cantos de pássaro são tocados, e reproduzi-la corretamente no painel.

Por exemplo, no primeiro painel (Figura 6.16), o utilizador ouve primeiro um canto grave, e depois um agudo. A resposta correta é, portanto, pressionar primeiro o botão inferior à esquerda do painel e depois o botão superior à direita.

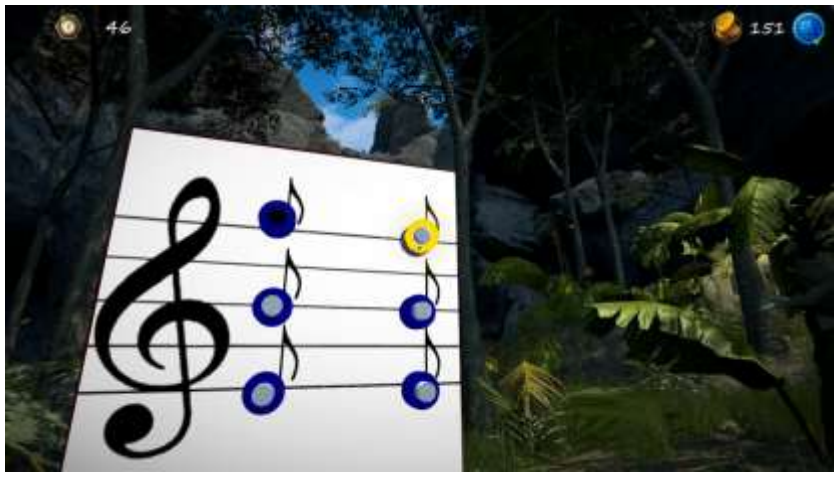


Fig. 6.16 Resposta do estudante no painel do Jogo dos Sons

Para selecionar um determinado botão, o utilizador tem apenas de colocar o cursor do rato sobre ele e este será realçado a amarelo. Se o botão pressionado for o correto, irá ficar verde; caso contrário, ficará preto.

O som é, normalmente, importante em qualquer jogo, mas é crucial neste, já que os jogadores têm de conseguir distinguir entre tons graves, agudos e médios, e, por isso, têm de estar muito concentrados.

O Ambiente de Jogo do Aluno pode ser acedido por todos os estudantes, mas destina-se sobretudo àqueles que têm entre 10 e 15 anos. O objetivo é promover a participação igualitária em atividades de programação, dentro e fora da sala de aula. Explora os conceitos de programação através de um ambiente virtual 3D que os estudantes podem explorar, de minijogos que ilustram conceitos relevantes, de uma plataforma colaborativa na qual podem

partilhar ideias para chegarem a determinada solução, de revisões feitas por outros que podem ajudar a ganhar perspetiva e aprofundar conhecimento sobre um determinado problema, e da comparação das suas respostas com aquela que foi introduzida pelo professor.

Poderão estar previstas futuras melhorias e atualizações para enriquecer o software do C4G com mais minijogos ilustrando outros conceitos de programação. A estrutura modular do jogo sério do Coding4Girls e a forma como foi desenvolvido pela Unity permitem precisamente este tipo de expansões futuras.

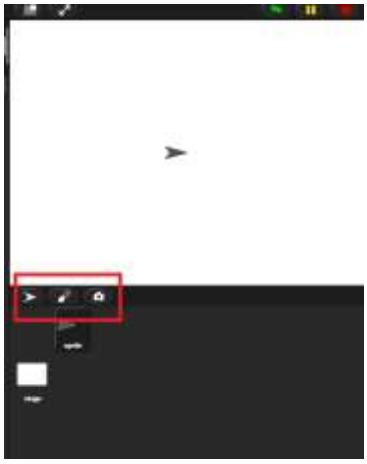
APÊNDICE 1. FICHAS DE APRENDIZAGEM

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.

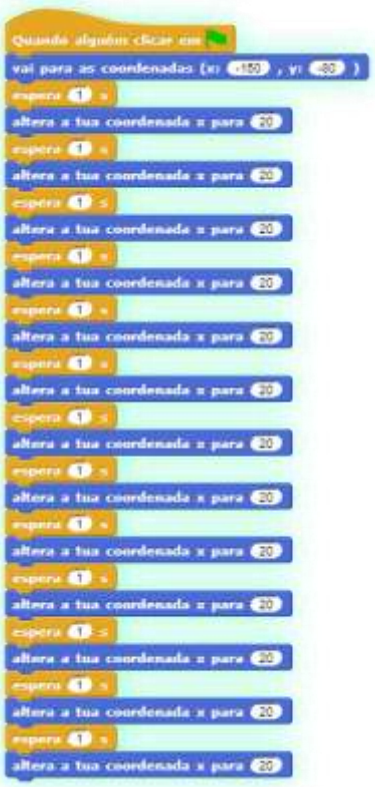
[Passo 7]

	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

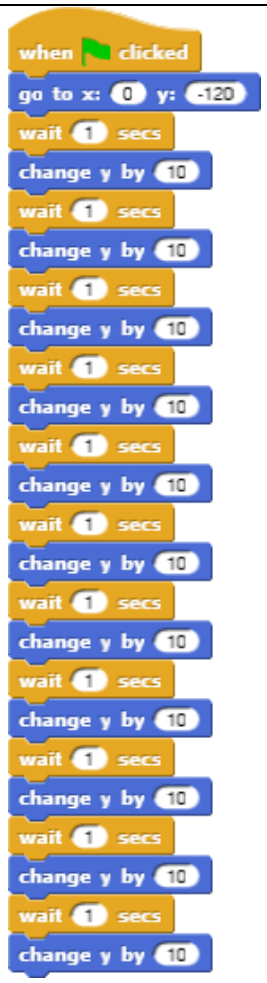
	de se movimentar e falar? Foi o mesmo Número de vezes que o resto da turma? Qual é a razão para tal?
Instrumentos e recursos para o Professor	Exemplo de programa: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home
Recursos/Materiais para os alunos	• Instruções 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.

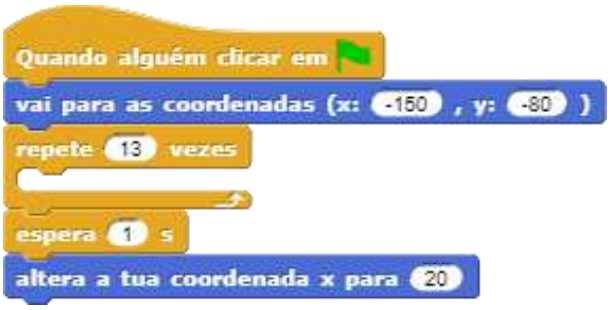
[Passo 3]

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

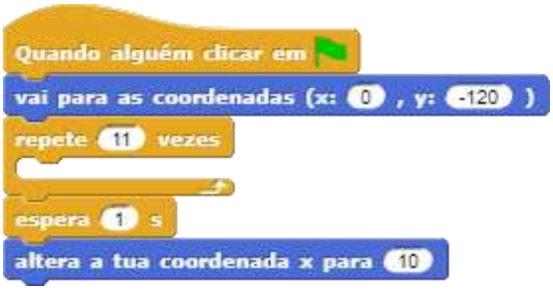
Há uma forma mais rápida de escrever este código, ordenando ao teu computador que o repita um certo número de vezes. Este é o *loop* "repete ____". Podes utilizá-lo quando uma mesma ação ou sequências de ações se repetem mais do que uma vez. Tenta repetir o código para ambas as tarefas, usando o *loop* 

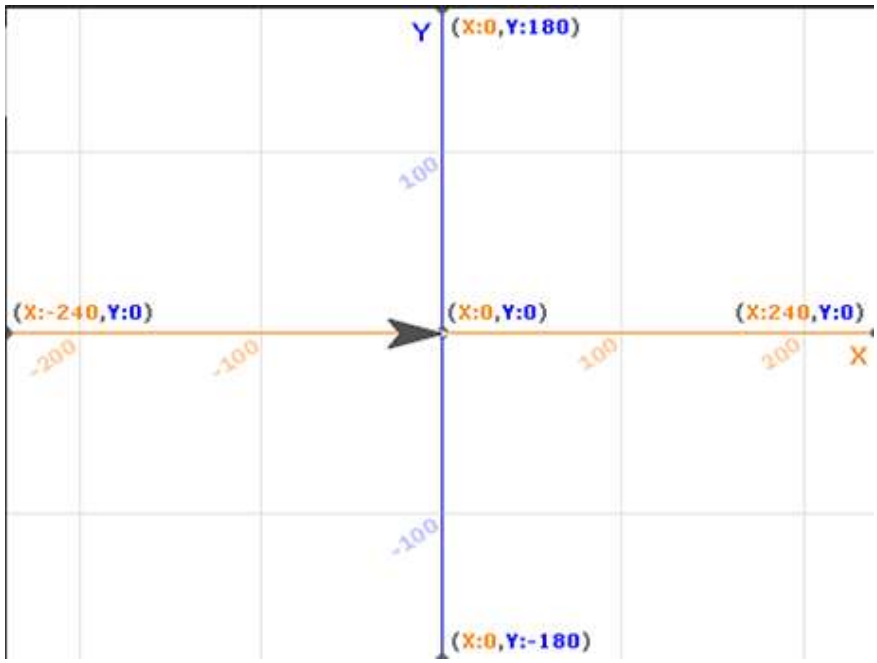
. O código que queres repetir tem de ser colocado dentro deste bloco, e tens de escrever no espaço em branco quantas vezes deve ser repetido.

Código para o cão:



Código para o macaco:

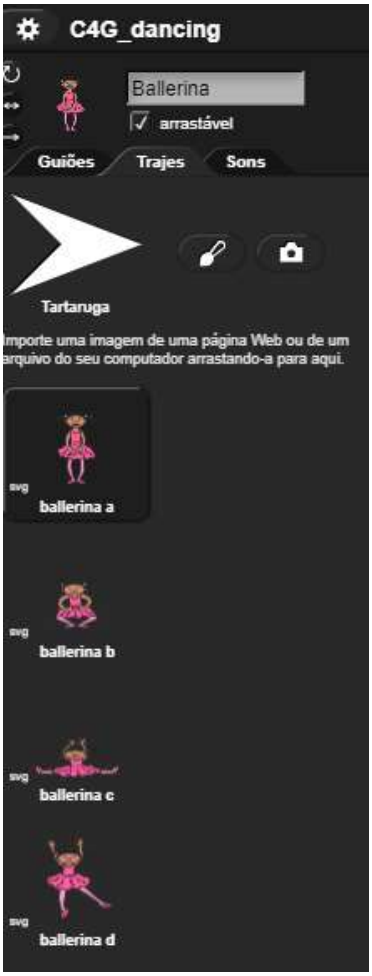


	Tarefa: Tenta fazer com que o cão corra até à bola e volte. Tarefa: Tenta fazer com que o macaco suba até ao topo da árvore e desça. Do que gostaste mais? Para ser mais fácil veres a posição do x e do y no *stage*, podes usar o cenário da grelha XY no Snap: 
Instrumentos e recursos para o Professor	● 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

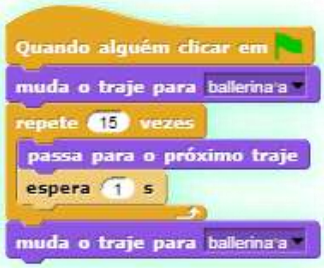
Título do cenário de aprendizagem	Mudar de traje e virar
Experiência prévia de programação	Movimento
Objetivos de aprendizagem	Objetivos gerais de aprendizagem: ● Construir uma sequência significativa de blocos Objetivos de aprendizagem específicos, orientados para o pensamento algorítmico: ● O estudante 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.

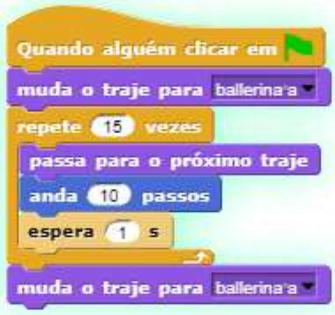
Solução:



[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]

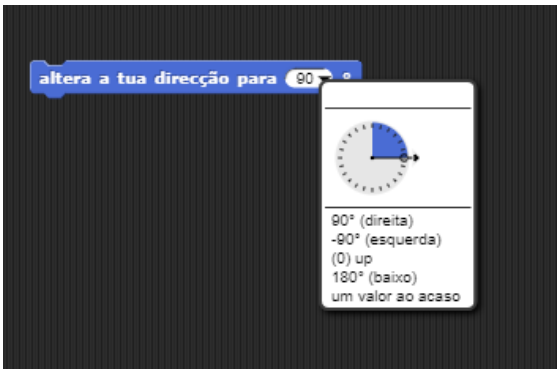
Abre um novo projeto e importa os trajes da Avery. Adiciona um

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 

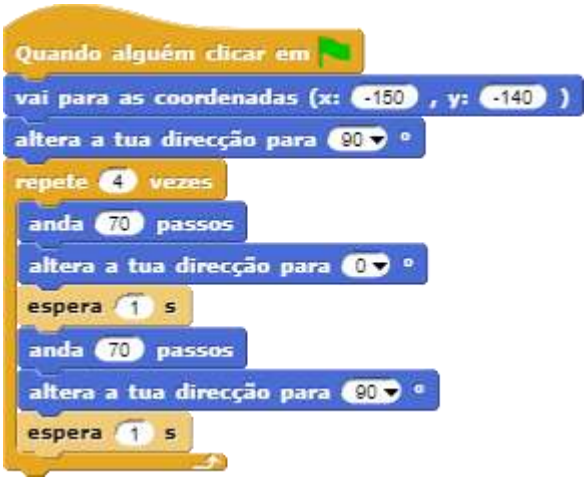
ou na direção contrária 

. 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.

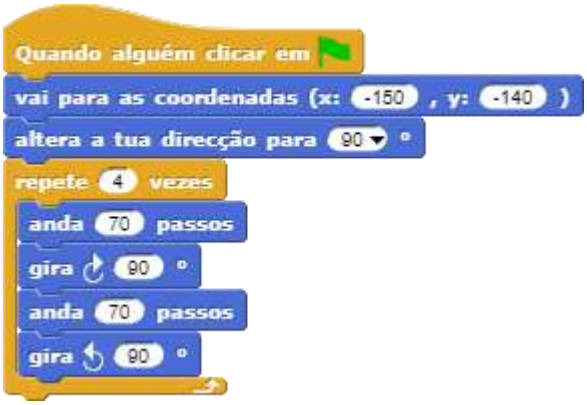
Abre

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:

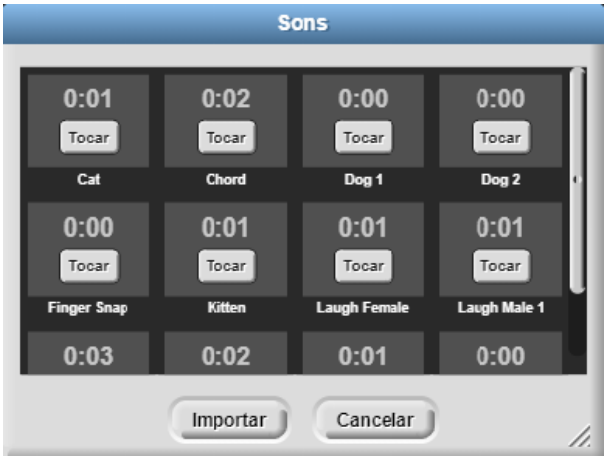
1. Importar um ficheiro de áudio da biblioteca do Snap;
2. Importar um som do próprio computador, arrastando-o até ao Snap!;
3. Gravar um novo som no Snap!

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).

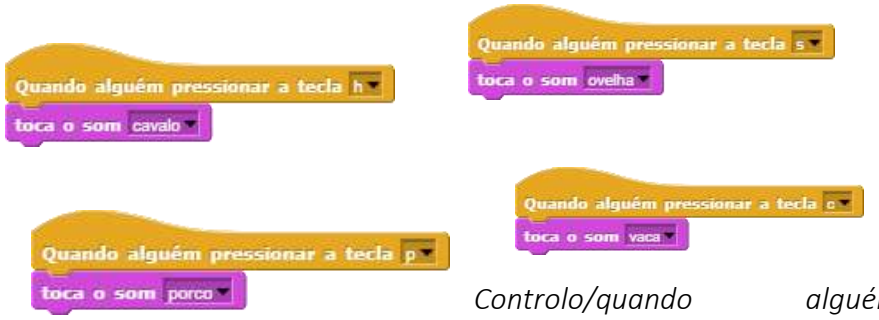


Os estudantes têm de programar o som do cão para ser reproduzido quando a tecla "D" for pressionada. Podem fazê-lo usando os blocos *Controlo/quando alguém pressionar [a tecla]* e *Som/toca o som [nome do som].*

[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]

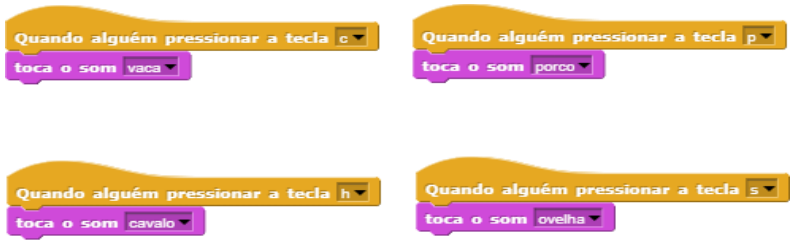
O próximo passo é programar a mensagem de boas vindas da camponesa. Quando o utilizador começar o jogo, a camponesa tem de dizer “Bem-vindo/a à minha quinta”. Em primeiro lugar, os estudantes têm de gravar esta mensagem. Para fazê-lo, usam o gravador de som (botão vermelho) localizado no separador de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

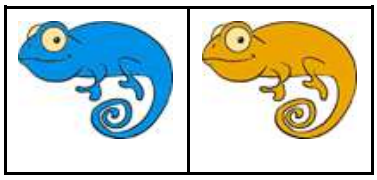
	novos *sprites* (camponês, galinhas, tratores…) e sons. **Reflexão e avaliação** Os estudantes devem resumir: a) Como adicionaram sons ao código; b) Quais foram os blocos utilizados para introduzir som no código; c) Quais foram os blocos de controlo que utilizaram no código; d) Como e quando utilizaram 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	**Breve descrição:** Programar um jogo simples em que o objeto irá

descrição das atividades	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?"*

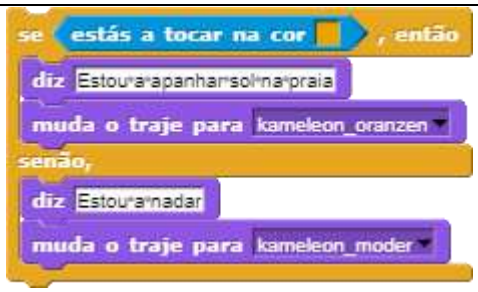
Existem duas formas:

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]

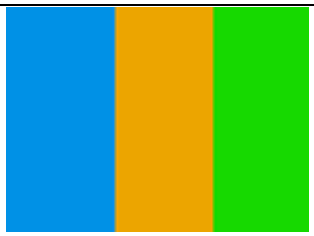
Para situações em que temos de executar certos comandos durante todo o programa, usamos o *loop* infinito. Tudo aquilo que fica sob o "*loop* infinito" irá ser executado repetidamente. Neste caso, podemos dizer aos alunos que é exatamente isto que pretendemos para criar este jogo.

[Código final]

[Versão completa]

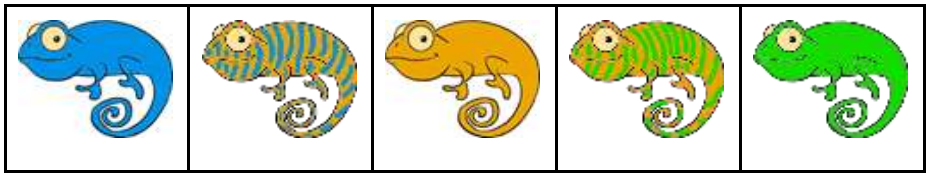
[Passo 1]

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

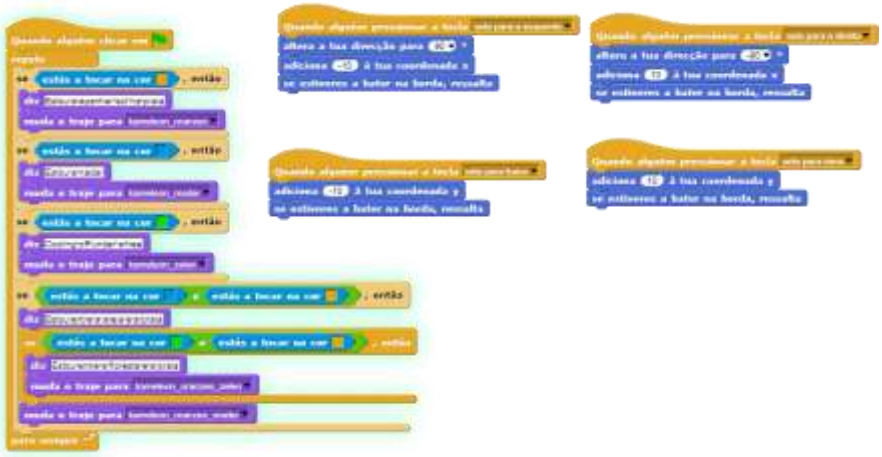
cores ao mesmo tempo:

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

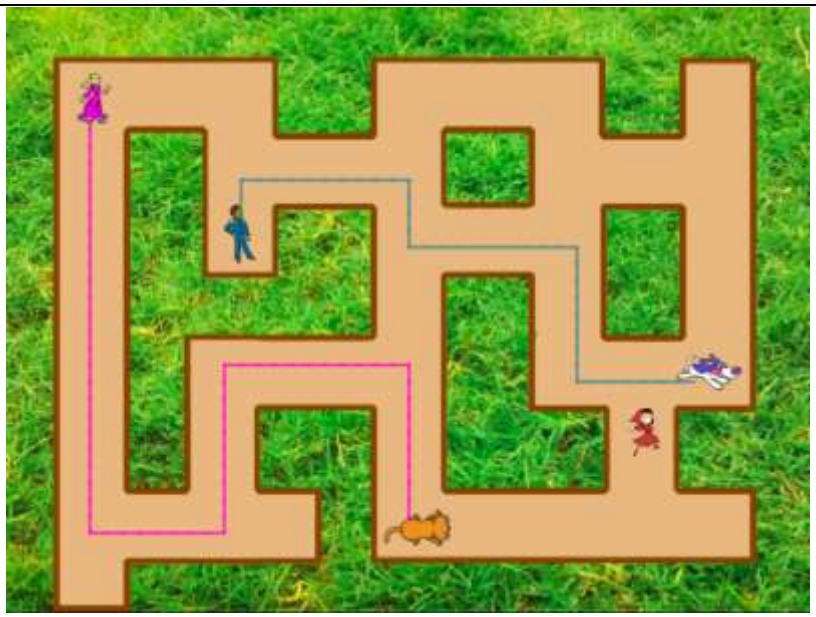
	código de movimento no template. Podem modificar as definições das teclas, de teclas de setas para personalizadas (WASD, por exemplo).  Para ajudá-los a compreender o conceito de *loop* infinito e perceber como o utilizar para verificar a cor do cenário, podes incluir código para detetar duas situações: 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 *sprites* 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]

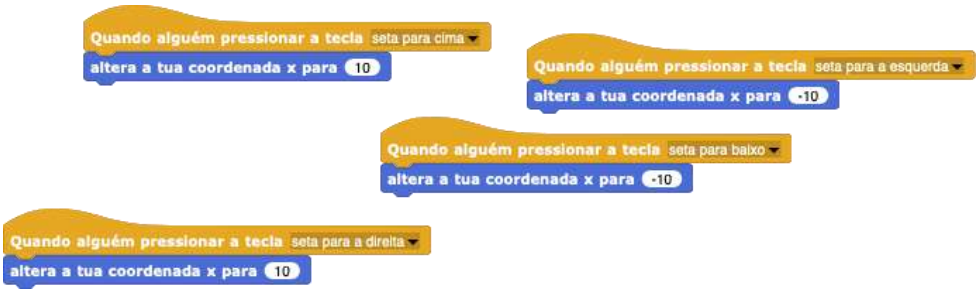
Os estudantes já têm a rapariga no início. Eles precisam encontrar outros quatro sprites (imagens) e colocá-los no labirinto. Para todos os sprites (imagens) deve-se definir o tamanho adequado (que deve ser menos que a largura dos caminhos do labirinto).

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]

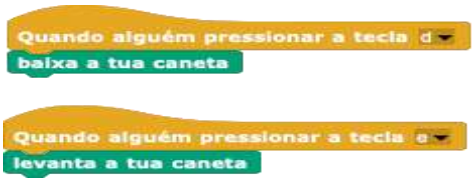
No próximo passo tem que se prevenir o movimento das raparigas pelo campo. Se faz isso adicionando uma barreira condicional ao tocar na cor castanha. Se a rapariga estiver a tocar na cor castanha (fim do caminho), ela volta 10 passos para trás. Nós não vemos aqueles dois passos e é como se a rapariga 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]

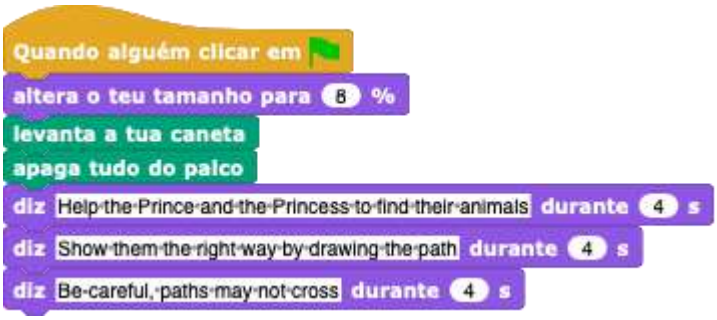
Por fim, os estudantes programam o comando “quando se clica na bandeira verde”, onde adicionam algumas instruções que a rapariga diz no início.

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:

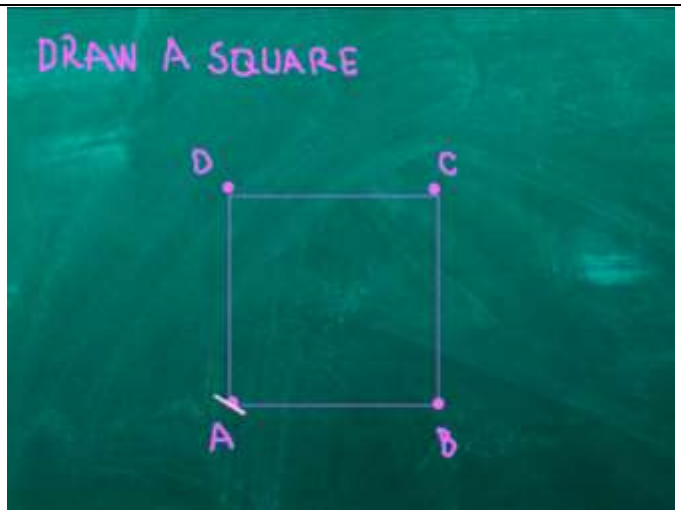
- Determine as coordenadas iniciais para o Príncipe e para a Princesa e escreva o código do movimento deles. Determine o tamanho

	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%20Prince%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 Aprendizagem	Desenhar com giz
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).

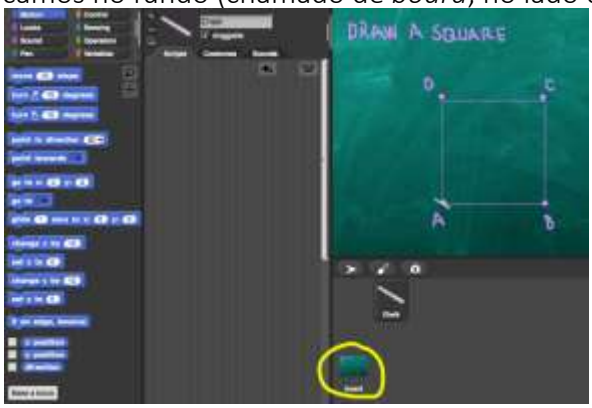
Por vezes, acontece pararmos o programa durante o jogo e então a

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*.

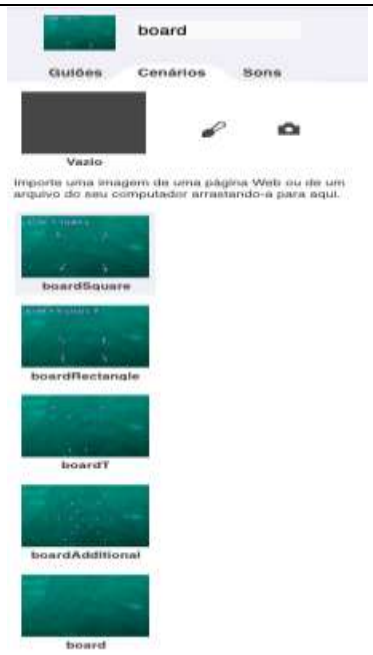
[Passo 3]

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).

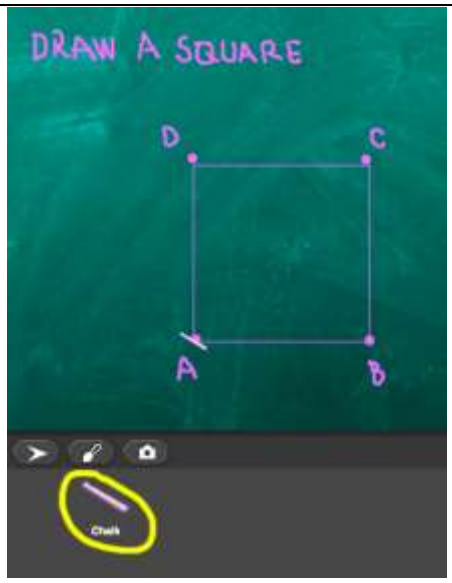


Ao clicar em *planos de fund*o podemos ver todos os três planos de fundo necessários (*boardSquare, boardRectangle, boardT*), já preparados para essa atividade.



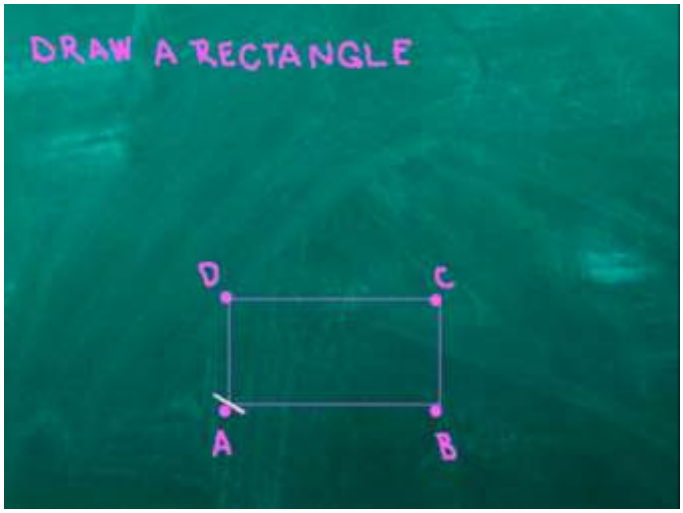
Para programar um código os estudantes devem clicar em *Scripts*. Para programar mudar um cenário escolhido escolhe-se um bloco de eventos quando a tecla R está pressionada e depois troca-se para personalizar *boardRectangle*.

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.

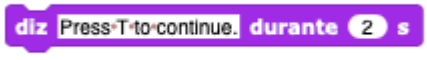
Muda-se a repetição de *loop*. Agora, o loop vai-se repetir 2 vezes.

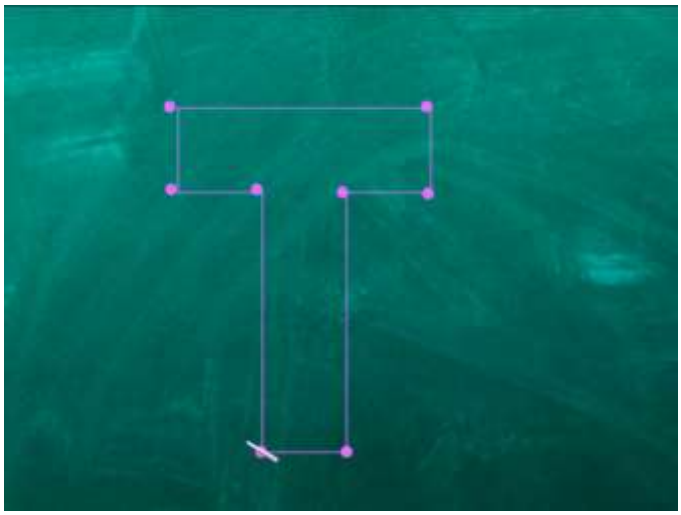
[Passo 5]

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".



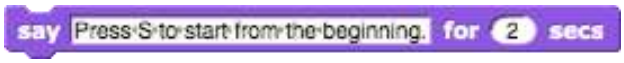
	[Step 6]  Depois de pressionar a tecla "T", o cenário muda para esse. Assim como no anterior, precisa-se conectar os pontos e desenhar a letra "T"” Os estudantes podem copiar o bloco de códigos anterior e corrigi-los. Os estudantes terão que mudar as coordenadas, que não são as mesmas que as anteriores. 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]

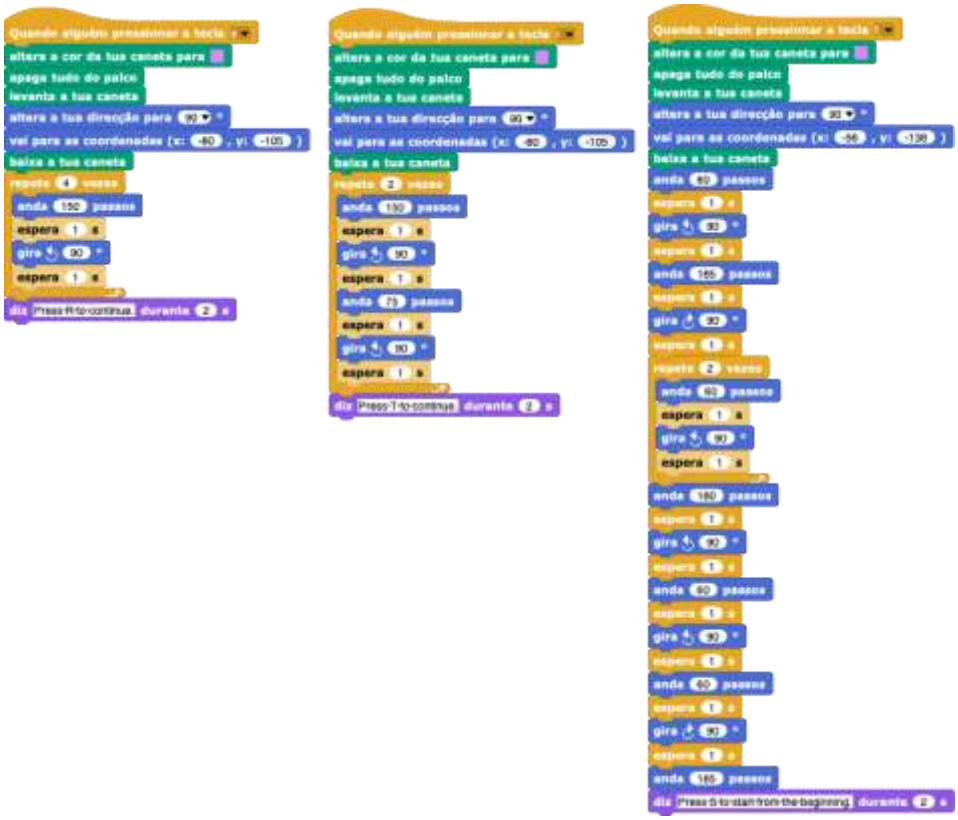
Uma vez em que mudamos o cenário, nós podemos retornar ao primeiro cenário para desenhar um quadrado. Então os estudantes terão que adicionar um último código. Deve-se repetir o [Passo 3/5].

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)

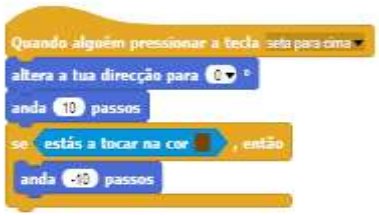
Cenário de Aprendizagem 9 - Apanhar o lixo e limpar o parque

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.

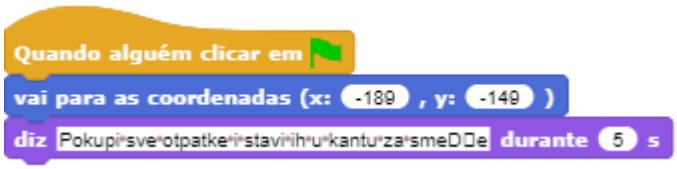
[Passo 1]

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]

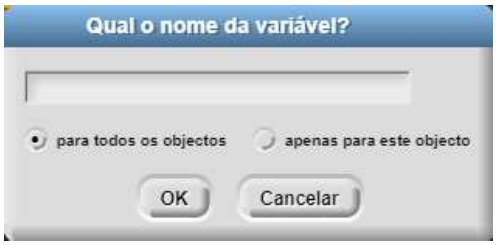
Para contar a quantidade de lixo que a rapariga recolheu, iremos usar variáveis.

O que é uma variável ?

A variável é como uma caixa onde guardamos algumas informações.

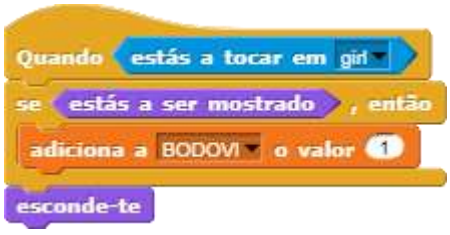
No nosso caso, conseguimos ver a nossa variável como uma caixa , designada por pontuação. Quando a rapariga recolhe um lixo, o lixo é guardado numa variável de pontuação. Esta variável conta quanto lixo a rapariga recolheu.

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*.

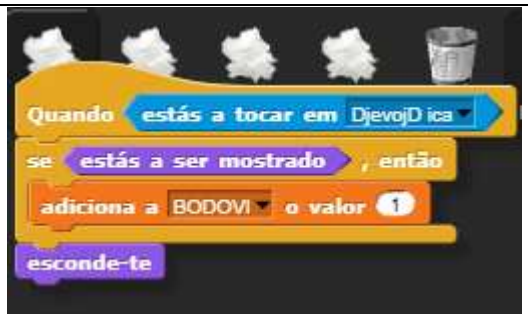
Agora, apenas selecione com o rato a nova garrafa e leve-a até a algum lado dentro do labirinto.

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



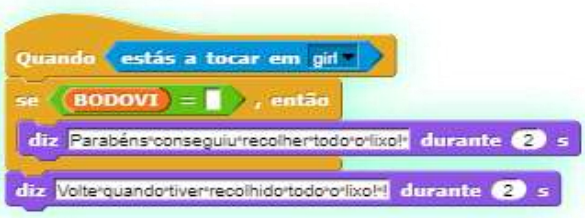
Podem repetir este passo para duplicar o bloco de código quando a Bandeira verde é selecionada – *mostrar*

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

[Passo 6]

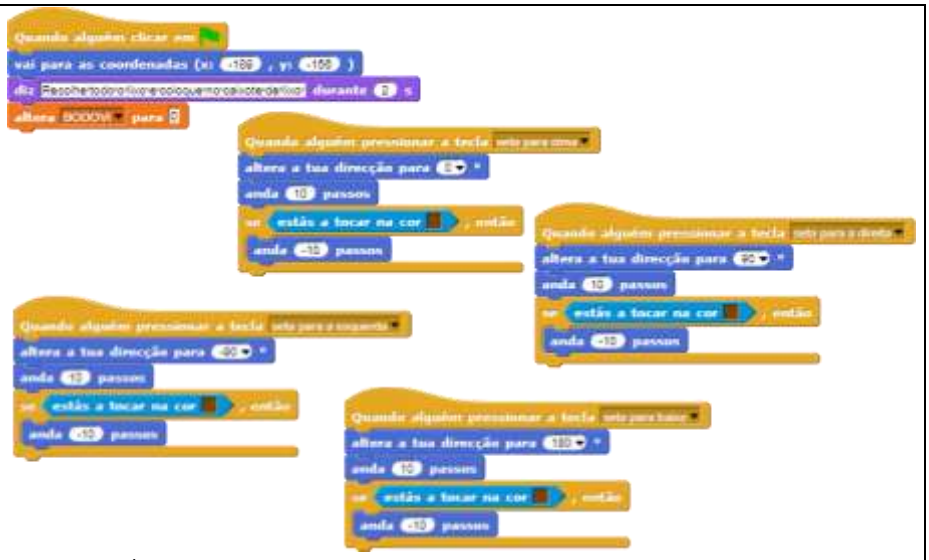
A última coisa que os alunos devem fazer é escrever um código para o caixote de lixo. A imagem do caixote já é dada, eles podem movê-la para qualquer lugar dentro do labirinto.

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.

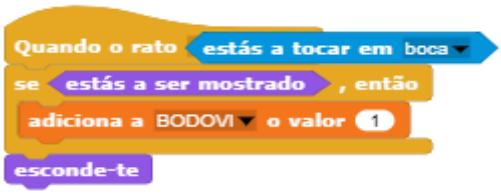


[Código Final]

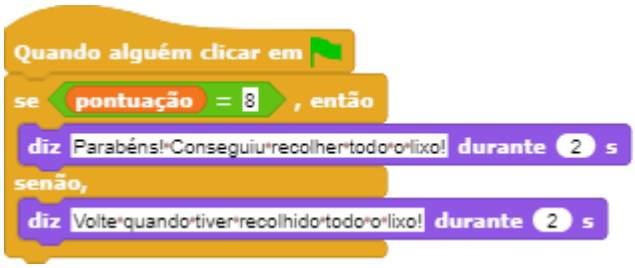
Rapariga



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".

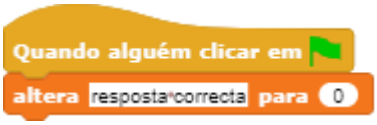
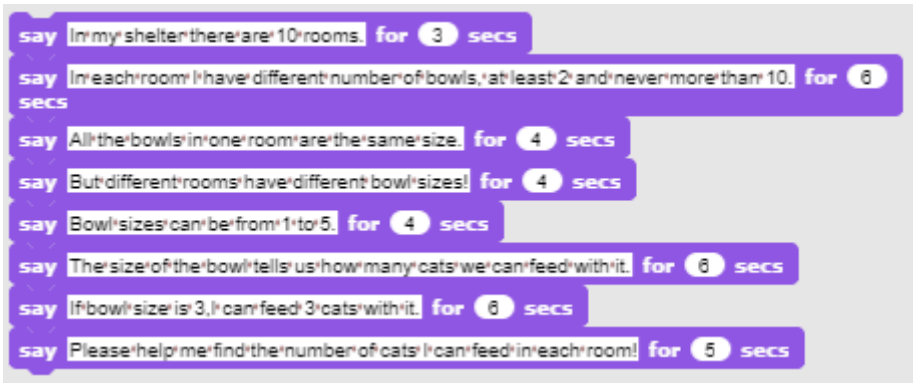
	● Se o jogador ou jogadora apanhar todo o lixo, o caixote poderá dizer: "Parabéns! Conseguiste recolher todo o lixo!" ● Se o jogador ou jogadora não recolher todo o lixo, o caixote poderá informar quais tipos de lixo ainda não foram recolhidos, por exemplo: " Ainda não recolheste todas as garrafas", "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

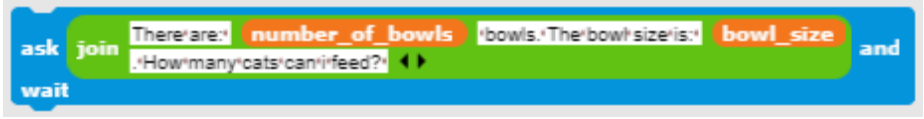
Título do cenário de Aprendizagem	Alimentar os Gatos
Experiência Prévia de programação	● Condições (bloco se , então, senão,) ● Imprimir o texto (bloco diz)
Objetivos de Aprendizagem	Objetivos gerais de aprendizagem: ● Definir e aumentar o valor da variável, ● Atribuir uma um valor da variável dentro/fora 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)

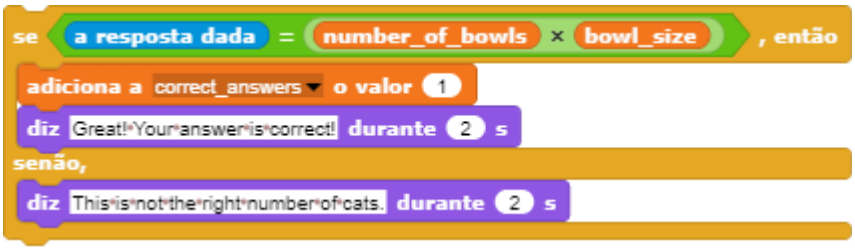
	para atribuir um número aleatório para a capacidade das tigelas (1-5). O contador das respostas corretas deverá ser definido inicialmente com o valor 0 e os outros dois não precisam ser definidos antes do loop/ciclo, porque iremos atribuir sempre um número aleatório a cada iteração do loop. Também queremos 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/alterações* 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.



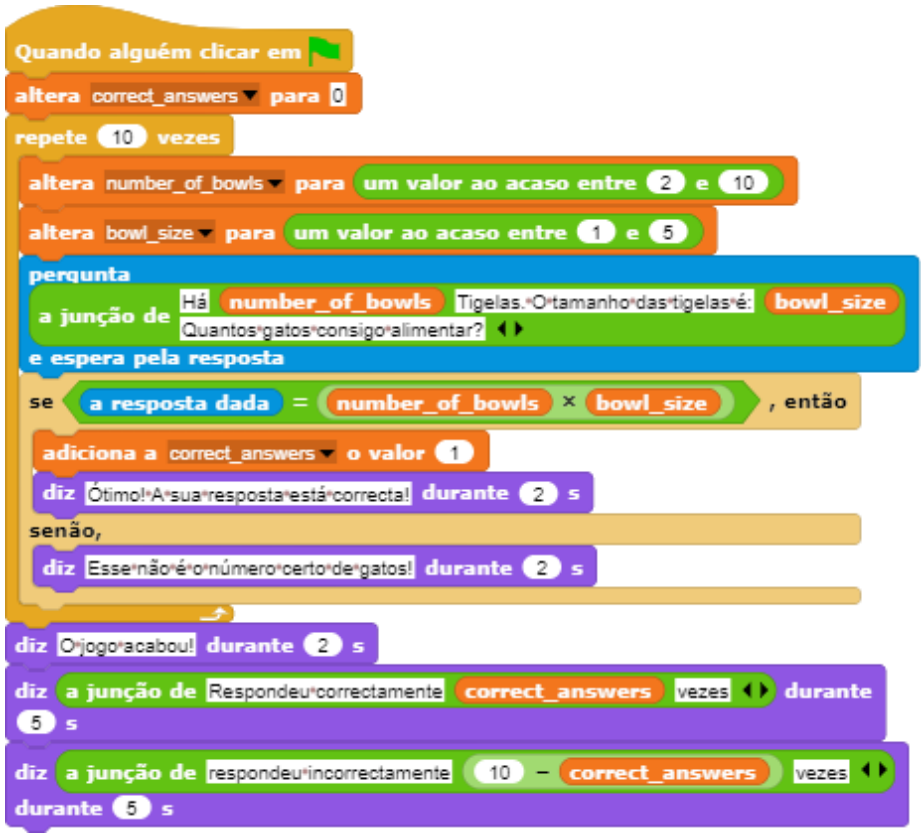
[Passo 11]

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

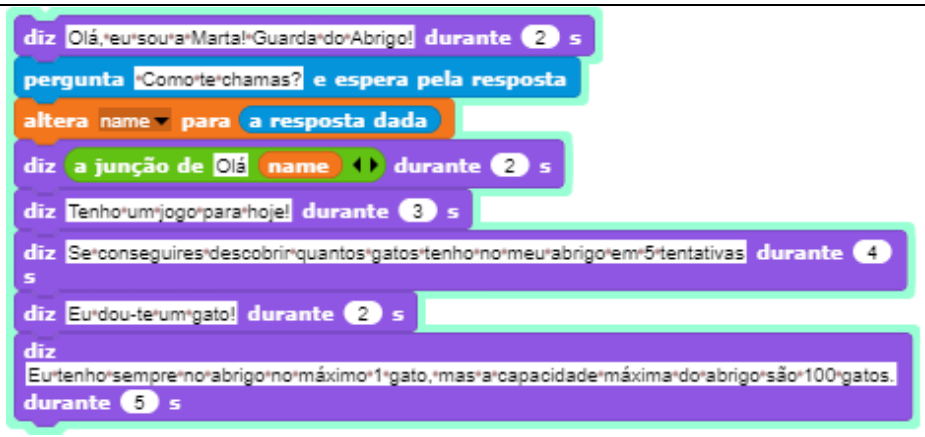
representará a guarda do abrigo.

[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.

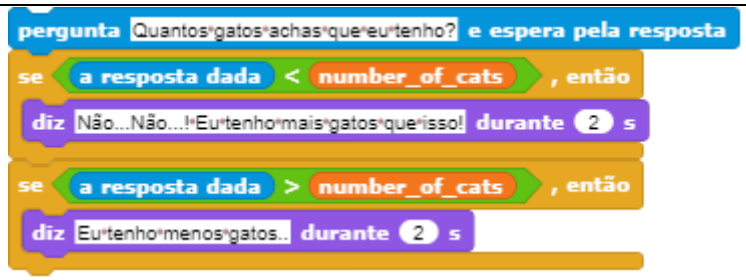


[Passo 5]

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 as cinco tentativas para acertar ou até pode vir a precisar de mais, é difícil 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 Lição	(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

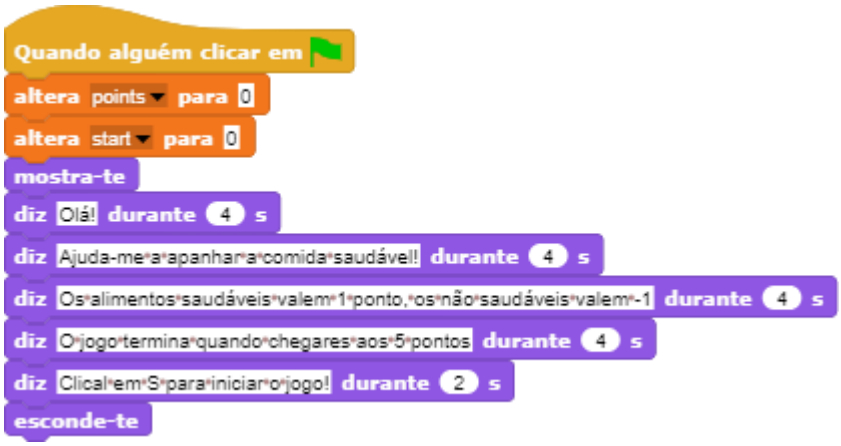
em pares. A professora deverá dar algumas dicas, explicando as partes mais complexas e ajuda no que for necessário.

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.

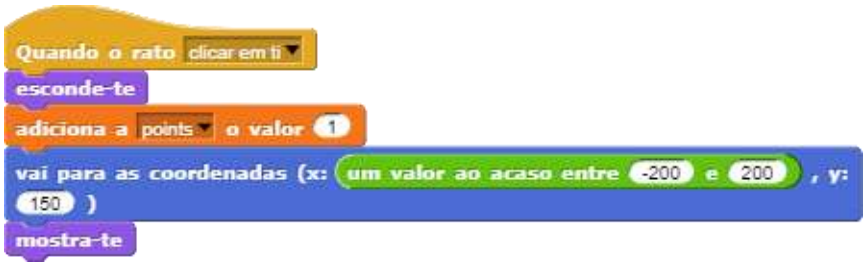
Quando a condição for verdadeira, ou seja, quando a pontuação for equivalente a 5, o *loop* para.

[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.

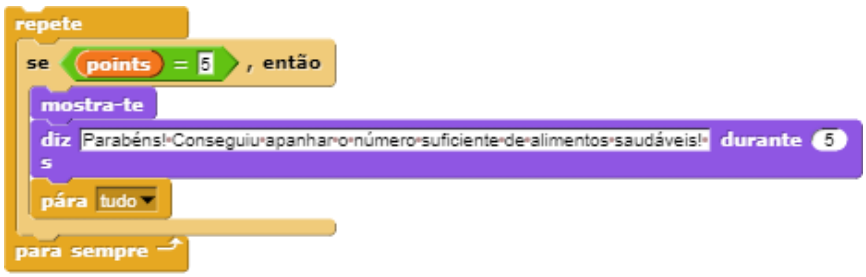


[Passo 7]

Vamos voltar a rapariga!

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

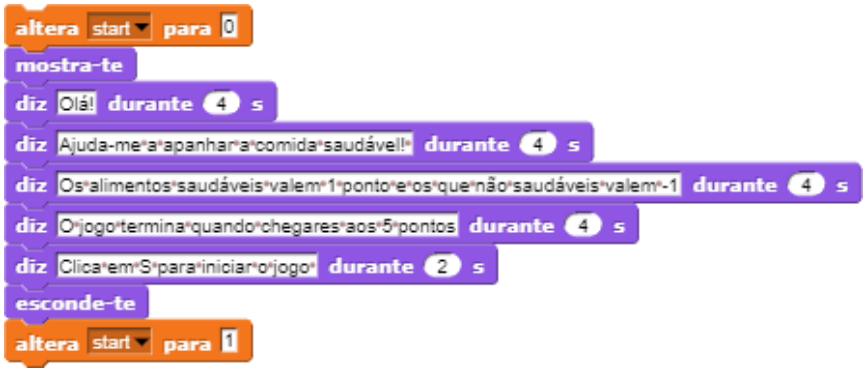
Precisaremos de um *loop* eterno, que irá verificar se o jogador(a) chegou aos cinco pontos. Se sim, a rapariga irá aparecer e dizer algo. Depois disso, adicionaremos o bloco *parar tudo*. E deixar os alunos descobrirem sozinhos o que isso significa (sem o sinal de parar a rapariga estará presente no 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]

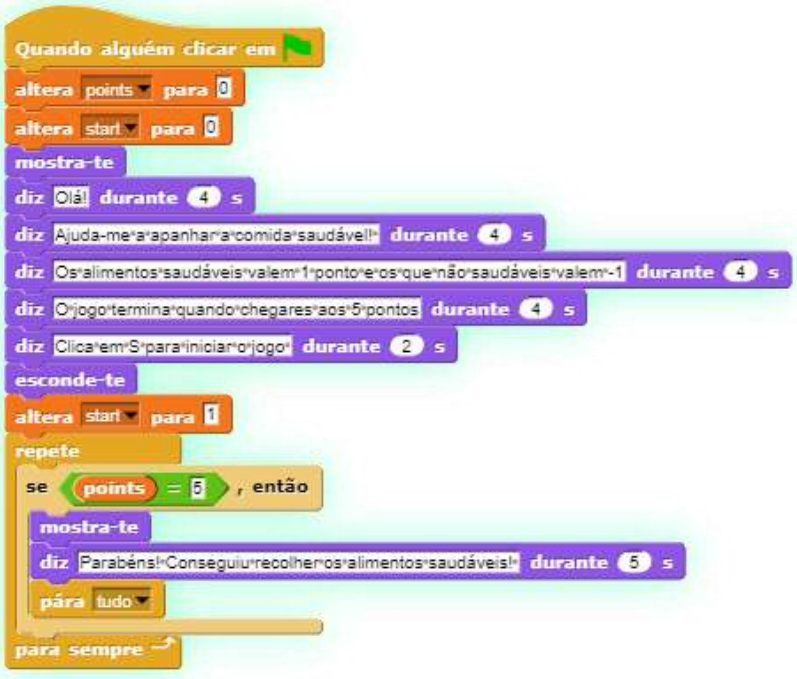
Os alunos agora podem duplicar a imagem da maçã muitas vezes e mudar o traje (se quiserem).

O código será o mesmo.

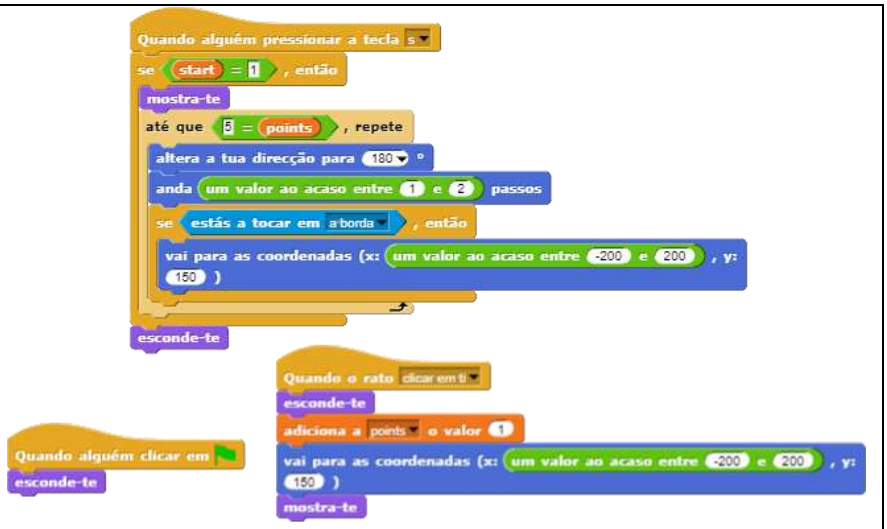
A única diferença é com a comida que não é saudável, onde eles perdem 1 ponto ao selecioná-la.

[Código Final]

Rapariga



Maçã



[Tarefas Adicionais]

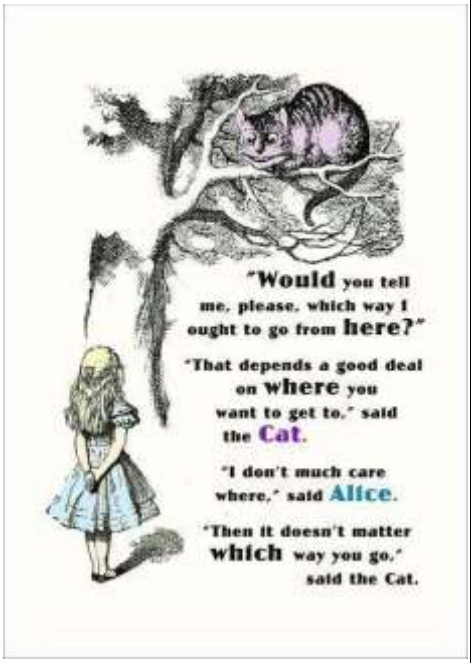
Os alunos podem adicionar tarefas adicionais de acordo com a sua vontade, e podem seguir as seguintes tarefas:

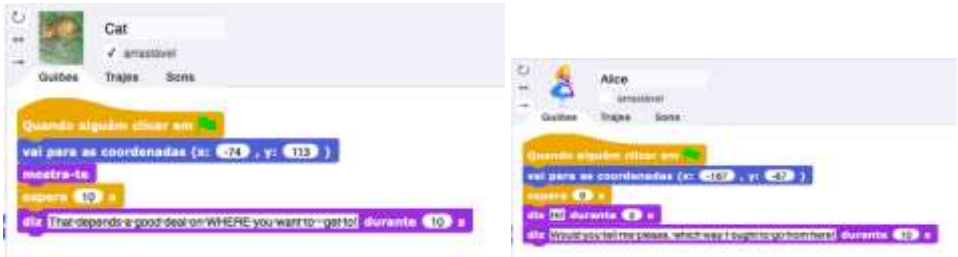
- Mudar o jogo para que a imagem da tigela apanhe o alimento.
- Adicionar uma nova imagem (uma tigela). É possível desenhar, encontrar online ou anexar uma fotografia para a imagem da tigela.
- Definir a posição de início (por exemplo: na parte superior do ecrã) e escrever o código para o movimento da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

Cenário de aprendizagem 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 dos sprites dentro da história ● Os estudantes usam *difunde a mensagem* para os diálogos entre sprites ● Os estudantes fazem com que os sprites se movam e mudem de tamanho ● Os estudantes sabem esconder e mostrar sprites ● Os estudantes refatoram e extendem o código dos sprites

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 no cenário com o título *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**

<table>
<tr><td>Duração das atividades</td><td>90 minutos</td></tr>
<tr><td>Estratégias e Métodos de ensino e aprendizagem</td><td>Aprendizagem ativa, aprendizagem baseada em game-design e resolução de problemas</td></tr>
<tr><td>Formas de Ensino</td><td>Trabalho individual/Trabalho a pares / Discussões presenciais</td></tr>
<tr><td>Sumário da Lição</td><td>(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:
 

<table>
<tr><td>Sprite</td><td>Atividade</td><td>Começar no início</td><td>Momento final</td><td>Duração</td></tr>
<tr><td>Coelho</td><td>Diz: Olá! Já ouviste falar da Alice e das</td><td>0</td><td>14</td><td>14</td></tr>
</table></td></tr>
</table>

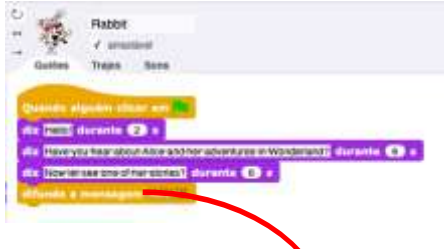
	suas aventuras no país das maravilhas? Agora vamos ver uma das suas histórias.			
Alice	Diz: *Poderia dizer-me, por favor, para qual caminho eu devo seguir a partir daqui*?	9	21	12
Gato	Diz: Isso depende de ONDE queres chegar!"	10	20	10

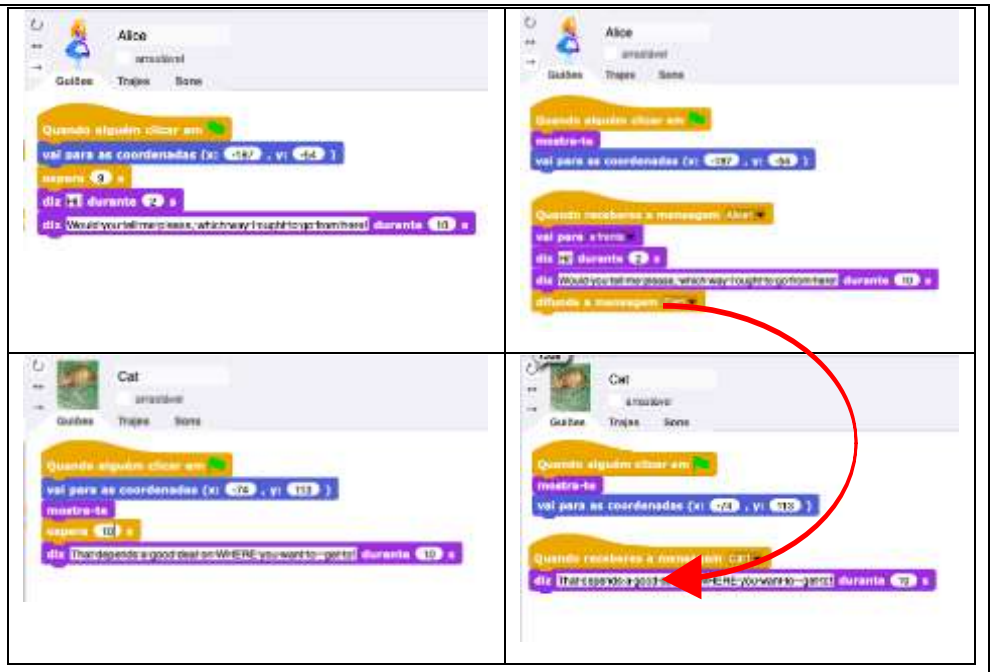
A conclusão é que sincronizar com o bloco espere por … segundos pode causar erros no comportamento dos personagens dentro do storytelling.

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.

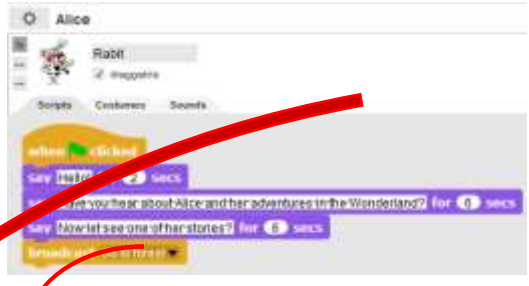
4.Os estudantes completam a história em pares.

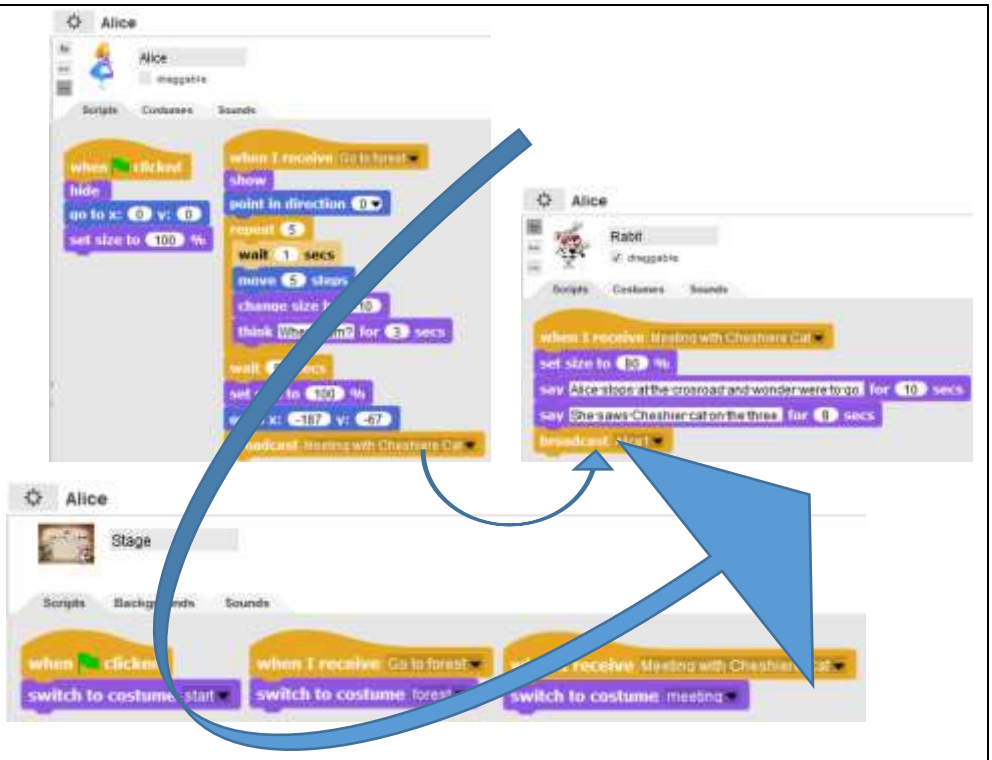
5. O professor comenta que contar histórias normalmente requer uma mudança nos trajes no stage. *"Vamos fazer a história da Alice mais completa ao iniciar a história do Coelho contrariamente ao contexto introdutório, movendo a ação para dentro da floresta onde a Alice está a andar e pergunta-se: "Onde 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.*

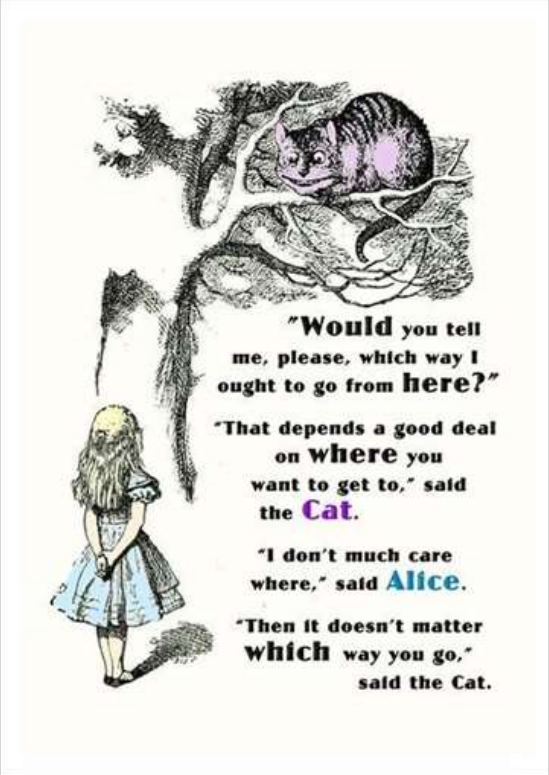
Quando ela atinge a junção, a cena muda com a mensagem "A encontrar o Gato Cheshire". Essa mensagem é recebida ao mesmo tempo pelo Coelho, que tem seu tamanho reduzido a 80% e ele continua a contar a história com seu tamanho reduzido.

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)

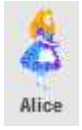
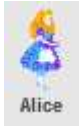
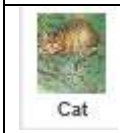
Appendix 1. Enredo da história/ Cenários

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*.	

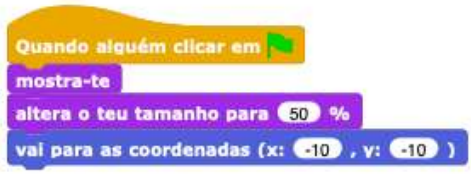
	Sem ação.Está visível desde a decoração anterior.	
	Recebe a mensagem*: Encontro com o Gato Cheshire.* Tamanho muda para 80% *Ele diz: "Alice para na encruzilhada e pergunta-se para onde ir. (por 10 segundos).* *Ele diz, "Ela viu o Gato Cheshire na árvore. " (por 8 segundos)* *Envia uma mensagem Alice1*	
	Recebe a mensagem *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]

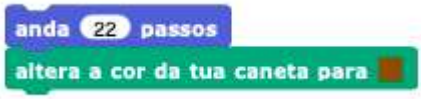
A imagem do lápis (sprite) deve receber mensagens “pinheiro” e “carvalho” e desenhar as árvores apropriadas em resposta a mensagem. Primeiramente, marque o lápis e adicione o código que capacitará desenhar o pinheiro usando uma caneta quando o sprite receber a mensagem “pinheiro”.

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 não 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.

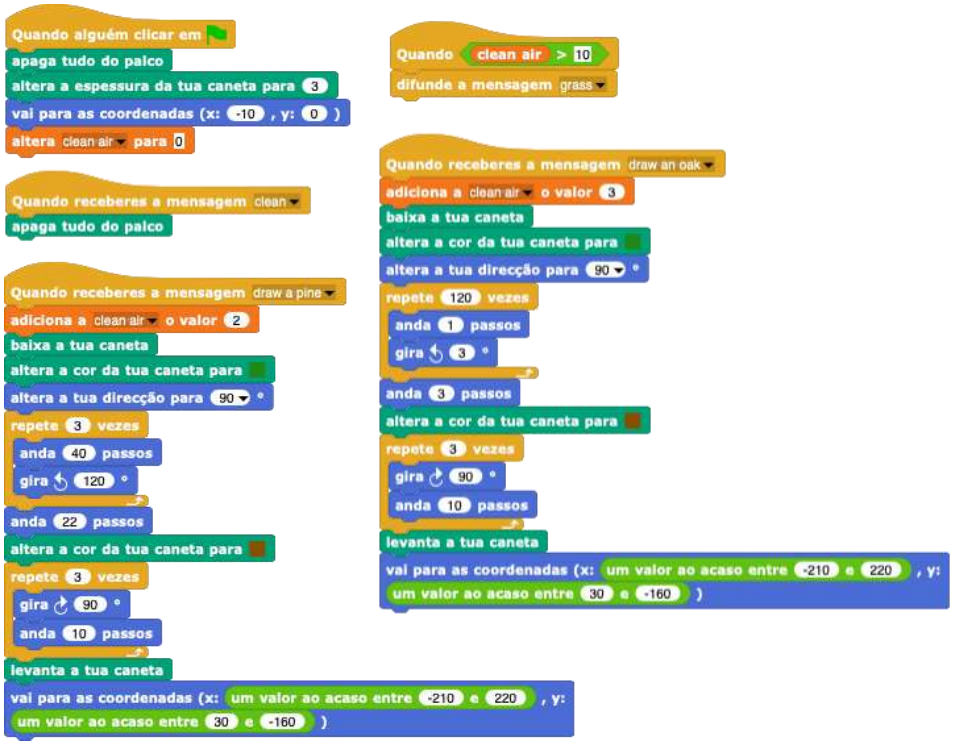
[Código final]

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.

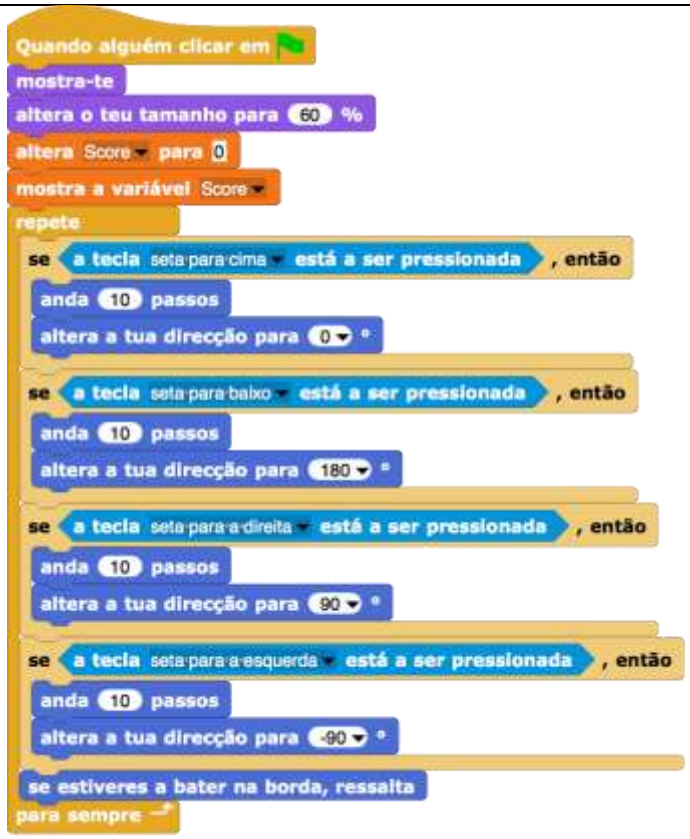
Implementação

[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.

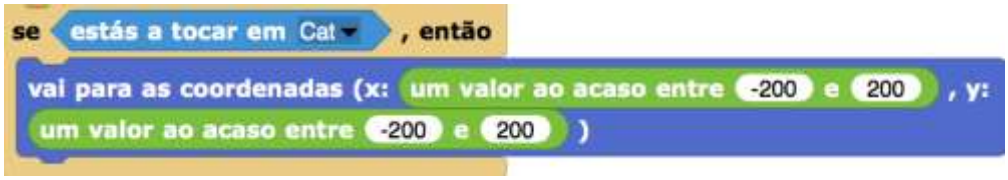
[Passo 2]

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



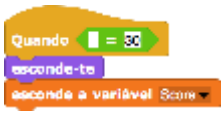
[Passo 3]

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]

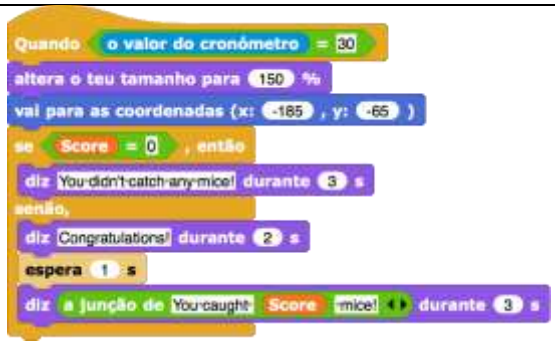
Os alunos determinam quando é que o jogo acaba. Eles fazem isso ao adicionar o cronómetro. Depois de algum tempo (ex.: 30 segundos) o rato e o gato desaparecem, a variável pontuação é escondida e o jogo termina.



Os alunos têm de adicionar esses guiões aos blocos do gato e do rato.

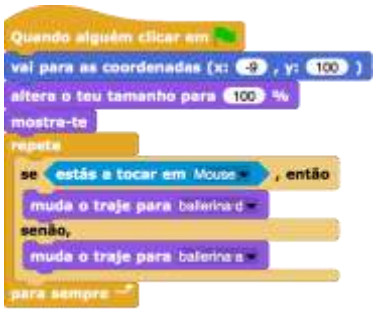
[Passo 7]

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

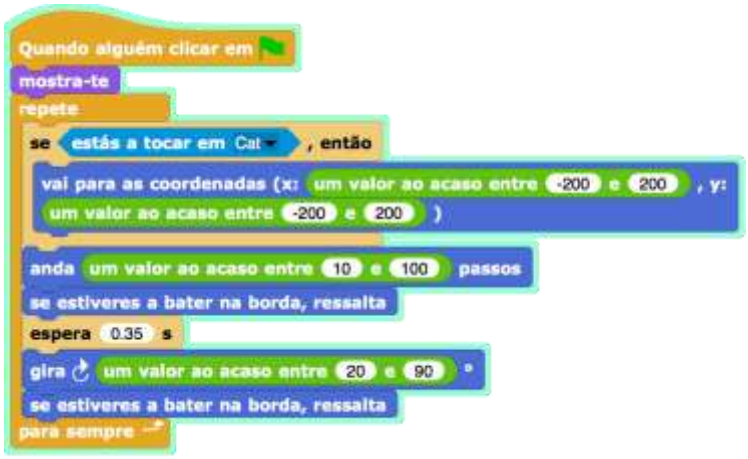
Os alunos ajustam o código:

- 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;

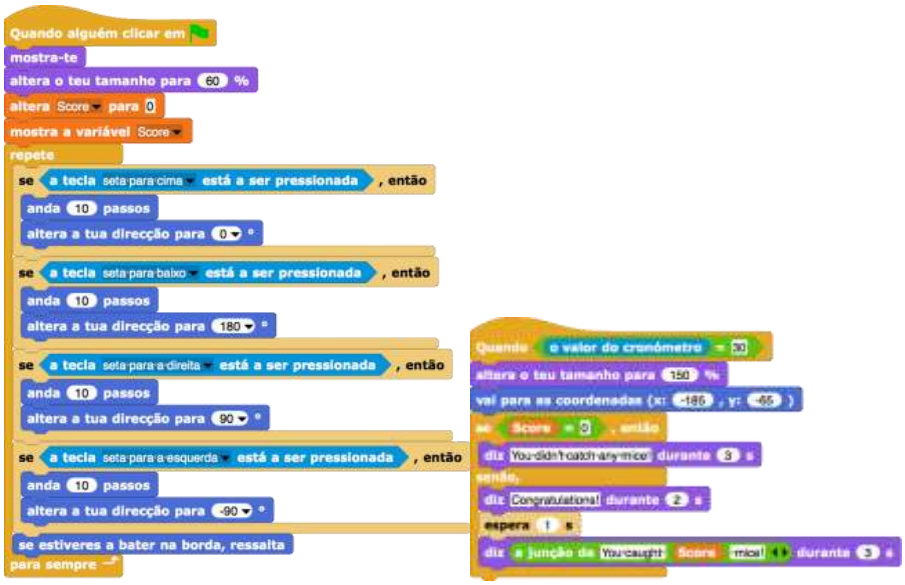
- etc;

[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: A rapariga vai fazer um piquenique e precisa de ajuda

Breve descrição das atividades	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]

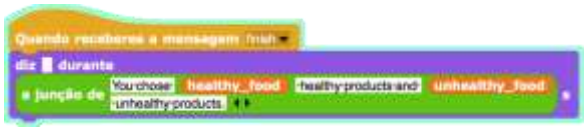
Quando o jogador acaba as suas compras, clica no botão Finish. Para dizer ao programa que o jogador clicou no botão (terminar de comprar comida), enviamos uma mensagem.

[Passo 6]

No fim, voltamos a imagem da rapariga.

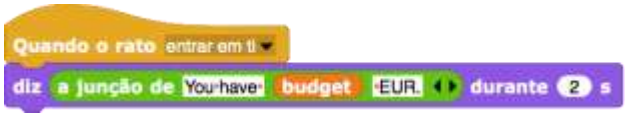
Quando o jogador termina as suas compras, queremos que a rapariga lhe diga quantos produtos saudáveis e não saudáveis ele comprou.

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:

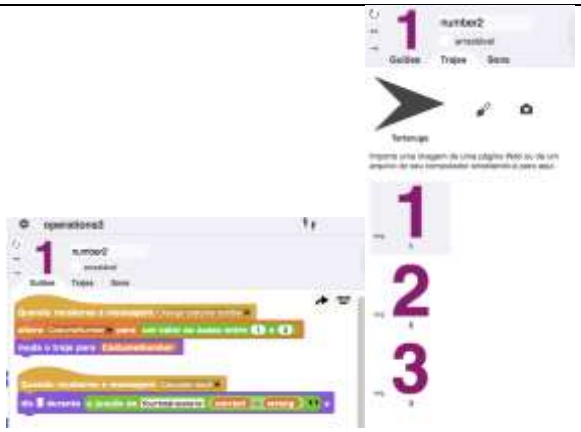
Breve descrição das atividades	Vamos ver enquanto jogamos se o jogador dominou as operações aritméticas no Snap!. As regras são as seguintes: Durante 10 vezes gerar operações aritméticas onde o primeiro operante possua 6 números aleatórios e o segundo operante seja um número entre 1 e 3. O jogador tem 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.

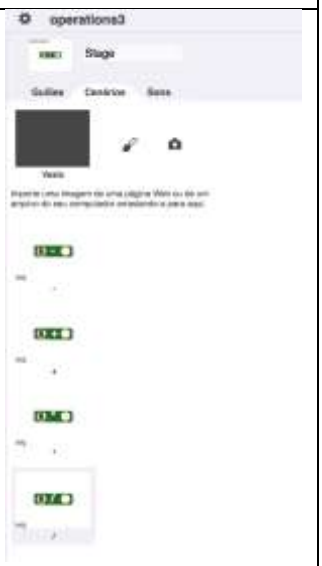
1. O professor discute como formular a condição da tarefa. A tarefa é formulada.

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 ● Blocos de sensores ● 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 o posicionamento dos *sprites* (numa tarefa adicional usam posicionamento aleatório).
Objetivos, Tarefas e	Breve descrição:

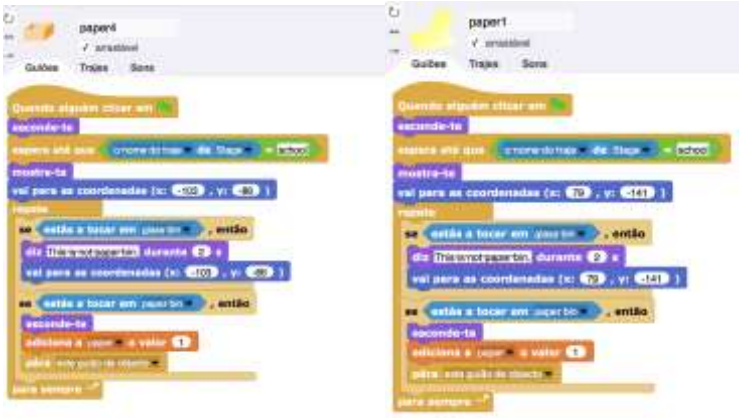
Breve descrição das atividades	Alguém deixou lixo em frente à escola. É pedido ao jogador para ajudar a organizar a recolha do lixo, ao separar na reciclagem o papel e o vidro. Quando o lixo é colocado no contentor correto, o lixo é escondido. Se o lixo for colocado no contentor errado, a mensagem - " 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

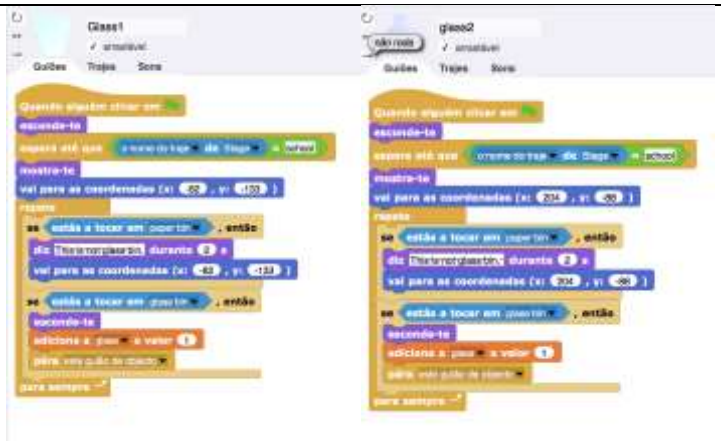
Sumário da lição	(Motivação-Introdução, Implementação, Reflexão e avaliação) 1. O professor expõe o problema da separação da recolha de lixo e comenta as cores dos contentores para os diferentes tipos de lixo - azul para o papel, verde para o plástico. 2. Diz aos alunos para jogar e descrever em palavras: 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

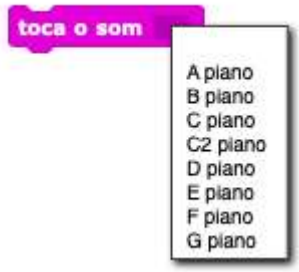
	 tângulo vermelho. ● Introduzir um novo *Sprite do Robô* e reduzir o número de elementos do stage no palco. ● Escrever o código apropriado do Robô para que possa entrar em diálogo com o jogador em vez do sprite do contentor blue.
Tools and Resources for the Teacher	Toda a atividade em Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling 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.

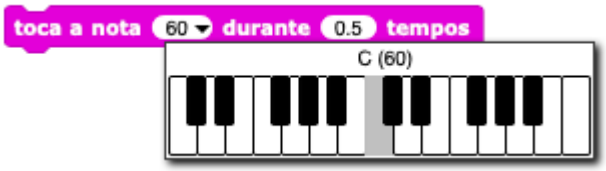
Os alunos testam o som de vários instrumentos.

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.

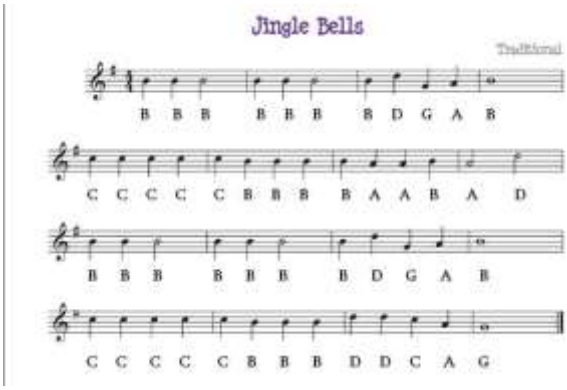


C	C#	D	Eb	E	F	F#	G	G #	A	Bb	B	C
60	61	62	63	64	65	66	67	68	69	70	71	72

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.

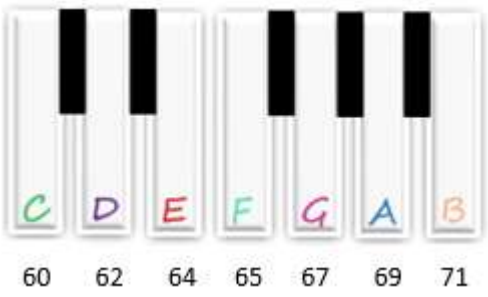


6. A tarefa é definida para reduzir o número de linhas no código que são repetidas. É discutido o comando a ser usado (repetir *loop*) Os alunos são divididos em equipas que têm que criar o jogo, definido no início da lição. Cada equipa discute o cenário do jogo e descreve o plano do jogo na folha de 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	Aprendizagem ativa, aprendizagem baseada em game-design, resolução de problemas.

ensino e aprendizagem	
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: 

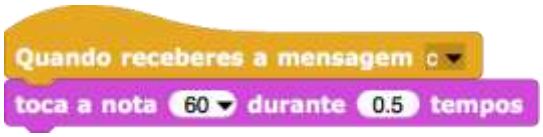
Fazer o mesmo para a tecla B e colocá-las no fim do teclado.

Desmarcar o botão "arrastável", para que os *sprites* das teclas não possam ser movidos enquanto está a tocar.

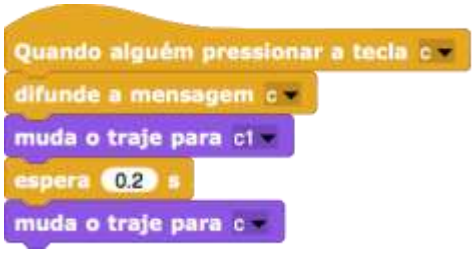
[Passo 2]

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”

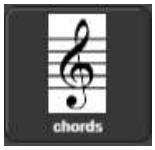
Se a tecla C no teclado estiver para baixo, o som irá repetir-se enquanto a tecla é pressionada. Isto acontece porque a mensagem “a” está em transmissão repetidamente. Para parar de transmitir uma mensagem, no fim do código adicionar o bloco “espera até que” do separador Controlo.

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]

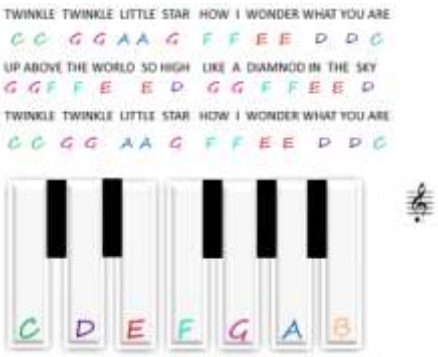
Criar um novo *sprite* e importar uma imagem de uma tecla de violino como traje. Este será um botão para exibir as palavras e as notas a serem tocadas.



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 Mu dar 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 Estratégias de Ensino e Aprendizagem	Aprendizagem ativa (discussões, experiência com um jogo previamente 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. 

	 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.

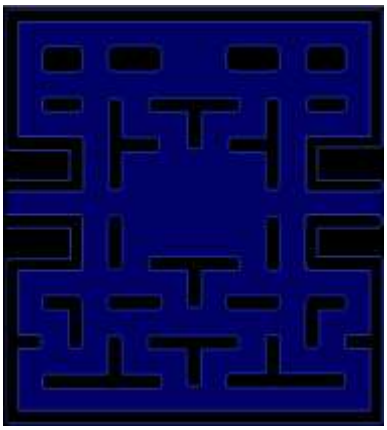
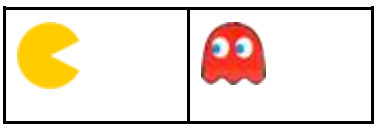
A forma como se desenha o cenário do stage para questões individuais é discutida.

	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:

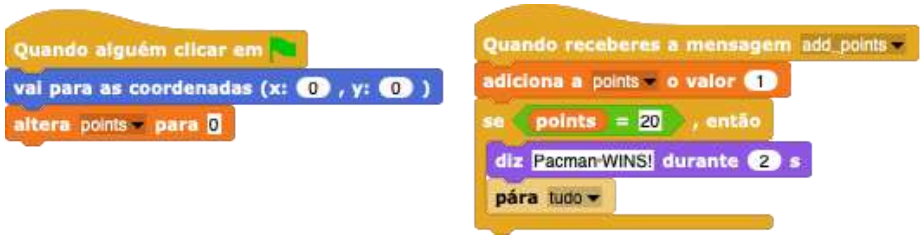
1. 0 graus - ACIMA
2. 180 graus - ABAIXO
3. 90 graus - DIREITA
4. 270 graus - ESQUERDA

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.

<table>
<tr>
<td></td>
<td>


[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.



</td>
</tr>
<tr>
<td>Instrumentos e recursos para o Professor</td>
<td>

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.

</td>
</tr>
</table>

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)

APÊNDICE 2. CÓDIGOS DOS CENÁRIOS DE APRENDIZAGEM

CENÁRIOS DE APRENDIZAGEM BÁSICOS

Cenários em Inglês

N.	TÍTULO	CÓDIGO
1	**Introduction to Snap! Interface** Students add a new sprite, add a costume to the sprite, edits the costume, and deletes one of them. Student creates a new background to the stage, edits it, and deletes unwanted ones.	starting
2	**Discover Snap! : move a spite** Help the students to discover the Snap! interface and code their first sprite so that it moves and speaks.	dispubeng
3	**Moving around the stage** Students learn how to move the sprite in x and y direction on the stage, code an easy program to solve the tasks given, learn how to turn the sprite in a different direction.	monkey
4	**Changing costumes and turning** Students learn how to change the sprite's costume to make an animation by changing between different types of rotation.	Dancer
5	**Sounds of the farm** Students learn how to program a simple game in which player can recognize the sounds of animals by pressing certain keys.	farm
6	**Chameleon's summer vacation** Program a simple game in which an object will change its costume based on the color of the background	chameleoneng
7	**Helping Prince and Princess to find their animals** A girl has to help the Princess find her cat and the Prince find	finding

	his dog by avoiding that the animals can meet each other during their path.	
8	Drawing with a chalk Students will be introduced into drawing different shapes with a code. They will learn to use loop repeat for shorten the code and to change a background.	chalk
9	Picking up trash and cleaning the park Students will learn how to use variables and how to duplicate a block of code or even a whole sprite.	cleaning
10	Feeding the cats Students will be introduced to the concept of multiple variable random value assignment inside a loop and how it is different from when we do it outside a loop. They will also learn about how to get, test and count correct player inputs.	feeding
11	Guessing the number of cats in a shelter Students will be introduced to "repeat until" loop and how to set the condition to implicitly track the condition that stops the game. They will also learn how to use variable in a different situations: to set a random value, as a counter or to get the players input.	shelter

Cenários em Esloveno

N.	TÍTULO	CÓDIGO
1	**Uvod v Snap!** Učenec doda nov lik, doda obleko, uredi obleko in izbriše obleko/lik. Učenec ustvari novo ozadje, ga uredi, zbriše odvečna ozadja.	uvod_slo
2	**Razišči Snap!: premikaj lik** To uro boš spoznal/a, kako z bloki likom naročimo naj se premikajo po odru in govorijo.	razisci_slo
3	**Premikanje po odru** Učenec se nauči, kako premikati lik v x in y smeri, kako napisati preprost program ter kako obrniti lik v drugo smer	premikanje_slo
4	**Menjava obleke in obrat** Učenec se nauči, kako liku zamenjati obleko in kako narediti animacijo z obračanji lika	obleka_slo
5	**Zvoki na kmetiji** Učenec se nauči programiranja preproste igre, v kateri lahko igralec s pritiskom na določene tipke prepozna zvoke živali	kmetija_slo
6	**Kameleon na počitnicah** Izdelaj preprosto igro, v kateri bo glavni objekt spreminjal svojo obleko glede na barvo ozadja na njegovi trenutni poziciji.	kameleon_slo
7	**Pomagaj princu in princeski najti svoje živali** Dekle mora pomagati princesi poiskati svojo mačko in princu svojega psa. Pri tem se mora izogniti srečanju med živalmi.	iskanje_zivali_slo
8	**Risanje s kredo** Učenec se seznani z risanjem različnih oblik s kodo. Nauči se uporabljati zanko ponovi za krajšanje kode ter kako zamenjati ozadje.	kreda_slo

9	**Pobiranje smeti in čiščenje parka** Učenec se nauči uporabljati spremenljivke in kako podvojiti blok kode ali celotno kodo lika.	ciscenje_parka_slo
10	**Nahrani mucke** Učenec se seznani s konceptom dodeljevanja več spremenljivk z naključno vrednostjo znotraj zanke in kakšna je razlika, če vrednost dodelimo zunaj zanke. Spoznali bodo tudi, kako pridobiti in preveriti igralčev odgovor ter kako šteti pravilne odgovore.	mucke_slo
11	**Mačje zavetišče** Učenec se seznani z zanko "ponavljaj dokler" in kako ustvariti pogoj za implicitno sledeje pogoju, ki konča z igro. Nauči se tudi, kako uporabljati spremenljivko v različnih situacijah: nastaviti naključno vrednost, kot števec ali kot igralčev vnos.	zavetisce_slo

Cenários em Italiano

N.	TÍTULO	CÓDIGO
1	**Snap!: introduzione** Lo studente impara ad aggiungere un nuovo sprite, a modifica il costume e ad eliminarne uno di essi. Impara a crea un nuovo sfondo per il palco, lo modifica e cancella quelli indesiderati.	Primipassi
2	**Scoprire Snap!** Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	Snap!
3	**Muoversi sullo "stage"** Il discente impara a spostare il suo Sprite in direzione x e y sullo "stage"; a preparare un semplice programma per risolvere i compiti assegnati; a far girare il suo Sprite in diverse direzioni.	stage
4	**Cambiare il costume e creare rotazioni**	ballerina

	Lo studente impara a cambiare il costume dello sprite per realizzare un'animazione.	
5	I suoni di una fattoria Gli studenti imparano come programmare un gioco semplice in cui il giocatore può riconoscere i suoni degli animali premendo determinati tasti.	fattoria
6	Vacanze estive di un Camaleonte Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	camaleonte
7	Alla ricerca degli animali del Principe e della Principessa! La ragazza deve aiutare la principessa a trovare il suo gatto e il principe a trovare il suo cane evitando che gli animali possano incontrarsi durante il loro.	labirinto
8	Disegnare con un gesso Gli studenti impareranno a disegnare forme diverse con un codice e ad usare la ripetizione ciclica per abbreviare il codice e cambiare lo sfondo.	gesso
9	Raccogliere la spazzatura e pulire il parco Gli studenti impareranno ad usare le variabili e a duplicare un blocco di codice o anche un intero Sprite.	spazzatura
10	Dare da mangiare ai gatti Gli studenti impareranno il concetto di assegnazione di più valori casuali, variabili all'interno e al di fuori di un ciclo. Impareranno anche come ottenere, testare e contare gli input corretti immessi dal giocatore.	gatto
11	Indovinare il numero di gatti in un rifugio Gli studenti impareranno ad utilizzare il ciclo "ripeti fino a" e ad impostare la condizione per interrompere il gioco. Impareranno anche ad usare la variabile in situazioni diverse.	rifugio

Cenários em Croata

N	TÍTULO	CÓDIGO
1	Uvod u sučelje alata Snap! Učenik dodaje nove objekte, dodaje kostime objektima, uređuje kostime te ih briše. Učenik stvara nove pozadine na pozornici, uređuje ju, te neželjene briše.	start_hr
2	Otkrijte Snap!: kretanje sprite Učenici otkriva gdje pronaći programske blokove i povezati ih u niz. Učenici će naučiti kako micati objekt i omogućiti da objekt govori.	disc_cro
3	Kretanje po pozornici Učenici će vidjeti kako izraditi program u kojem će se objekt kretati u smjeru x, te izraditi program u kojem će se objekt kretati u smjeru y.	kretanje
4	Mijenjanje kostima i okretanje Učenik uči kako promijeniti kostim objekta te kako napraviti animaciju. Također uči kako se između njih može mijenjati različite vrste rotacije objekta.	ples
5	Zvukovi s farme Učenici će se upoznati kako programirati jednostavnu igru u kojoj igrač može prepoznati zvukove životinja pritiskom na određene tipke.	farma
6	Kameleonov ljetni odmor Učenici će biti upoznati s blokom naredbi osjeta boja i kako ga koristiti u logičkim izrazima da bi razlikovali dinamično promjenjiva stanja igre i dali prave odgovore.	chamele on_cro
7	Pomaganje princu i princezi u pronalasku njihovih životinja Učenici će se upoznati s crtanjem pokretom tipke. Uz to će naučiti kako koristiti uvjete kojima će spriječiti da se objekt kreće po cijelome ekranu.	finding_ hr

8	Crtanje s kredom Učenici će se upoznati s crtanjem različitih oblika pomoću kôda. Naučit će koristiti petlju ponavljaj kako bi smanjili veličinu kôda i promijeniti pozadinu.	kreda
9	Skupljanje otpada i čišćenje parka Učenici će se upoznati kako koristiti varijable i kako kopirati blok kôda ili čak cijeli objekt.	park
10	Hranjenje mačaka Učenici će biti upoznati sa konceptom dodjele slučajne vrijednosti varijabli unutar petlje te će razlikovati kada navedeno napravimo van petlje. Naučiti će kako dobiti, testirati i izbrojati ispravne unose igrača.	hranjenje
11	Pogađanje broja mačaka u skloništu Učenici će se upoznati s petljom ponavljaj dok i kako postaviti uvjet koji zaustavlja igru. Također će naučiti kako koristiti varijable u različitim situacijama: za pohranjivanje nasumične vrijednosti, kao brojač ili za pohranjivanje vrijednosti koju upiše igrač.	pogodi

Cenários em Búlgaro

N.	TÍTULO	CÓDIGO
1	Увод в Snap! Ученикът добавя нов спрайт, добавя костюм към спрайта, редактира костюма и изтрива едино от двете. Ученикът създава нов фон на сцената, редактира го и го изтрива.	start_bg
2	Да разучим Snap!: Движещ се спрайт Помогнете на учениците да разучат интерфейса на Snap! и да съставят код за първия им спрайт, така че той да се движи и говори.	dissnap_bg
3	Движейки се по сцената ченикът се научава как да движи спрайта в x и y посока на сцената, създава лесна програма за решаване на задачите и се научава как да завърти спрайт в различна посока.	monkey_bg
4	Смяна на костюми и завъртане Ученикът се учи как да промени костюма на спрайт при завъртане, за да направи анимация.	dancer_bg
5	Звуци от фермата Учениците научават как да програмират игра, в която играчът може да разпознава звуците на животните чрез натискане на определени клавиши.	sounds_bg
6	Лятната ваканция на хамелеона Програмирайте проста игра, в която обект да променя костюма си в зависимост от цвета на фона.	cham_bg
7	Помогнете на принца и принцесата да намерят своите домашни любимци Момиче трябва да помогне на принцесата да намери котката си, а на принцът да намери кучето си, като трябва да	find_bg

	избегне възможността животните да се срещнат по пътя.	
8	Рисуване с тебешир Учениците ще бъдат запознати с рисуването на различни фигури(форми) използвайки код. Те ще се научат да използват цикъл с повторение за намаляне обема на кода и за промяна на фона.	draw_bg
9	Събиране на боклук и почистване на парка Учениците ще научат как да използват променливи и как да копират блок с код или дори цял спрайт.	clean_bg
10	Нахранете котките Учениците ще бъдат въведени в концепцията за присвояване на случайно число на няколко променливи в цикъл, както и каква е разликата при използването им вътре и извън цикъл. Те също така ще научат как за да зададат, тестват и изброят правилно въведеното от играча.	cats_bg
11	Познайте броя на котките в приюта Учениците ще бъдат запознати с цикъл "repeat until" и как да зададат условието, при което спира играта. Те също така ще се научат как да използват променливи в различни ситуации: за задаване на произволна стойност, като брояч или като стойност въведена от играча.	shelter_bg

Cenários em Grego

N.	TÍTULO	CÓDIGO
1	Εισαγωγή στο Snap! Ο μαθητής προσθέτει ένα νέο στοιχείο, προσθέτει ένα κοστούμι στο στοιχείο, επεξεργάζεται το κοστούμι και διαγράφει ένα από αυτά. Ο μαθητής δημιουργεί ένα νέο φόντο στη σκηνή, το επεξεργάζεται και διαγράφει τα ανεπιθύμητα.	start_gr
2	Ανακαλύψτε το Snap!: μετακινήσε μία εικόνα Βοήθησε τους μαθητές να ανακαλύψουν την πλατφόρμα προγραμματισμού Snap! και να κωδικοποιήσουν την πρώτη τους εικόνα ώστε να κινείται και να μιλάει.	dissnap_gr
3	Ας μετακινηθούμε γύρω από τη σκηνή Ο μαθητής μαθαίνει πώς να μετακινεί ένα στοιχείο σε κατεύθυνση x και y στη σκηνή, δημιουργεί ένα εύκολο πρόγραμμα για την επίλυση των καθηκόντων του, μαθαίνει πώς να γυρίζει το στοιχείο σε διαφορετική κατεύθυνση.	monkey_gr
4	Αλλαγή κοστουμιών και στροφή Ο μαθητής μαθαίνει πώς να αλλάζει το κοστούμι του στοιχείου για να κάνει ένα κινούμενο σχέδιο αλλάζοντας μεταξύ διαφορετικών τύπων περιστροφής.	dancer_gr
5	Ήχοι φάρμας Οι μαθητές μαθαίνουν πώς να προγραμματίζουν ένα απλό παιχνίδι στο οποίο ο παίκτης μπορεί να αναγνωρίζει τους ήχους των ζώων πατώντας συγκεκριμένα κουμπιά.	sounds_gr
6	Καλοκαιρινές διακοπές του Χαμαιλέοντα Προγραμματίστε ένα απλό παιχνίδι στο οποίο ένα αντικείμενο θα αλλάξει το κοστούμι του με βάση το χρώμα	cham_gr

	του φόντου.	
7	**Βοηθώντας τον Πρίγκιπα και την Πριγκίπισσα να βρουν τα ζώα τους** Το κορίτσι πρέπει να βοηθήσει την Πριγκίπισσα να βρει τη γάτα της και τον Πρίγκιπα να βρει τον σκύλο του, αποφεύγοντας τα υπόλοιπα ζώα που μπορεί να συναντήσουν κατά τη διάρκεια της πορείας τους.	find_gr
8	**Ζωγραφίζοντας με κιμωλία** Οι μαθητές θα μάθουν να ζωγραφίζουν διαφορετικά σχήματα με κώδικα. Θα μάθουν να χρησιμοποιούν βρόχους επανάληψης για να συντομεύσουν τον κώδικα και να αλλάξουν φόντο.	draw_gr
9	**Μαζεύοντας σκουπίδια και καθαρίζοντας το πάρκο** Οι μαθητές θα μάθουν πώς να χρησιμοποιούν μεταβλητές και πώς να αντιγράφουν ένα κομμάτι κώδικα ή ακόμα και ένα ολόκληρο στοιχείο.	clean_gr
10	**Ταΐζοντας τις γάτες** Οι μαθητές θα μάθουν την έννοια της πολλαπλής ανάθεσης τυχαίας τιμής σε μεταβλητές μέσα σε ένα βρόχο και πώς αυτό είναι διαφορετικό αν το κάνουμε εκτός βρόχου. Θα μάθουν επίσης για τον τρόπο λήψης, δοκιμής και μέτρησης των σωστών εισόδων από τον παίκτη.	cats_gr
11	**Μαντέψτε τον αριθμό των γατιών στο καταφύγιο** Οι μαθητές θα μάθουν το βρόχο «επανάληψη έως» και πώς να ρυθμίσουν τη συνθήκη ώστε να παρακολουθούν τη συνθήκη που σταματά το παιχνίδι. Θα μάθουν επίσης πώς να χρησιμοποιούν τη μεταβλητή σε διαφορετικές καταστάσεις: για να ορίσουν μια τυχαία τιμή, ως μετρητή ή για να πάρουν είσοδο από τον παίκτη.	shelter_gr

Cenários em Português

N.	TÍTULO	CÓDIGO
1	**Introdução à interface Snap!** Os alunos adicionam um novo sprite, adicionam um traje ao sprite, editam o traje e excluem um deles. O aluno cria um novo plano de fundo para o palco, edita-o e exclui os indesejados.	starting_PT
2	**Descubra o Snap! : mover um Sprite** Os alunos descobrem a interface do Snap! e codificam o seu primeiro sprite para que ele se mova e fale.	dispubeng_PT
3	**Movendo-se pelo palco** Os alunos aprendem como mover o sprite nas direções x e y do palco, codificam um programa fácil para resolver as tarefas dadas e aprendem a virar o sprite em uma direção diferente.	monkey_PT
4	**Mudando de traje e girando** Os alunos aprendem a mudar o traje do sprite para fazer uma animação e alternam entre os diferentes tipos de rotação.	dancer_PT
5	**Sons da quinta** Os alunos aprendem a programar um jogo simples no qual o jogador pode ouvir os sons dos animais pressionando certas teclas.	farm_PT
6	**Férias de verão do camaleão** Os alunos programam um jogo simples em que um objeto muda de traje de acordo com a cor do fundo	chameleoneng_PT
7	**Ajudando o príncipe e a princesa a encontrar seus animais** Uma menina tem que ajudar a Princesa a encontrar seu gato e o Príncipe a encontrar seu cachorro, evitando que os animais se cruzem durante o caminho.	finding_PT

8	Desenho com giz Os alunos aprendem a desenhar formas diferentes com um código. Aprendem a usar ciclos para encurtar o código e alterar um plano de fundo.	chalk_PT
9	Recolher o lixo e limpar o parque Os alunos aprendem a usar variáveis, a duplicar um bloco de código e um sprite.	cleaning_PT
10	Alimentando os gatos Os alunos aprendem o conceito de atribuição de valores aleatórios a múltiplas variáveis dentro de um loop e como é diferente de quando o fazemos fora de um loop. Eles também aprendem a obter, testar e contar as entradas corretas do jogador.	feeding_PT
11	Adivinhando o número de gatos em um abrigo Os alunos aprendem o ciclo "repetir até" e como definir a condição para rastrear a condição que interrompe o jogo. Aprendem ainda a usar variáveis em diferentes situações: definir um valor aleatório, como um contador ou para obter dados dos jogadores.	shelter_PT

Cenários em Turco

N.	TÍTULO	CÓDIGO
1	Snap! ara yüzünü tanıyalım Öğrenciler yeni bir karakter (sprite/kukla) ekler, karaktere bir kostüm ekler, kostümü düzenler ve bunlardan birini siler. Öğrenciler sahne için yeni bir arka plan oluşturur, düzenler ve	başla

	istenmeyenleri siler.	
2	Snap!'i keşfet - Karakteri hareket ettirme Öğrencilerin Snap! arayüzünü keşfetmelerine ve ilk kodlarını oluşturarak hareket eden ve konuşan bir karakter oluşturmalarına yardımcı olun.	keşfet
3	Sahnede hareket etme Öğrenciler sahnede karakteri x ve y yönünde hareket ettirmeyi öğrenir, verilen görevleri çözmek için basit bir program programlar, karakteri farklı yöne çevirmeyi öğrenir.	hareket
4	Kostüm değiştirme ve dönüşler Öğrenci, farklı rotasyon türleri arasında geçiş yaparak bir animasyon yapmak için karakerin kostümünü nasıl değiştireceğini öğrenir.	dans
5	Çiftlikteki sesler Öğrenciler, oyuncunun belirli tuşlara basarak hayvanların seslerini tanıyabileceği basit bir oyun programlamayı öğrenirler.	çiftlik
6	Bukalemenun yaz tatili Bir nesnenin kostümünü arka planın rengine göre değiştireceği basit bir oyun programlayın	bukalemun
7	Prens ve Prensese evcil hayvanlarını bulmaları için yardım et Kız, seyahatleri sırasında hayvanların birbirleriyle karşılaşmasını engelleyerek Prenses'in kedisini bulmasına ve Prens'in köpeğini bulmasına yardım etmelidir.	prenses
8	Tebeşir ile çizim yapmak Öğrencilere bir kodla farklı şekiller çizme tanıtılacaktır. Kodu kısaltmak ve arka planı değiştirmek için döngü tekrarını	çizim

	kullanmayı öğrenecekler.	
9	**Çöpleri toplamak ve parkı temizlemek** Öğrenciler değişkenleri nasıl kullanacaklarını ve bir kod bloğunu veya hatta bütün bir hareketli grafiği nasıl kopyalayacaklarını öğrenecekler.	çöpler
10	**Kedileri besleme** Öğrencilere, bir döngü içinde çok değişkenli rastgele değer atama kavramı ve bir döngü dışında yaptığımızdan nasıl farklı olduğu anlatılacaktır. Ayrıca doğru oyuncu girdilerinin nasıl alınacağını, test edileceğini ve sayılacağını da öğrenecekler.	kediler
11	**Barınaktaki Kedi Sayısını Tahmin Etmek** Öğrencilere "----e kadar tekrar et" döngüsü ve oyunu durduran koşulu örtük olarak izlemek için koşulun nasıl ayarlanacağı tanıtılacaktır. Ayrıca değişkeni; rastgele bir değer belirlemek, sayaç olarak veya oyuncuların girdisini almak gibi farklı durumlarda nasıl kullnacaklarını da öğrencekler.	barınak

CENÁRIOS DE APRENDIZAGEM AVANÇADA

Cenários em Inglês

N.	TÍTULO	CÓDIGO
12	**Catching healthy food** Students will learn how to randomly move for X steps and choose a position and also how to use variables and conditionals for preventing other event.	food

13	Storytelling Students will learn how to plan storytelling, how to use broadcast messages for synchronisation of the activities of sprites and stage changes.	storytelling
14	Drawing Student will learn to draw in Snap!, to change the colour and the pen thickness, and how to use variables and conditionals that cause a new event.	drawing
15	Catch the mouse Student will be introduced to the concept of multiple variable random value assignment. They will learn how to use the Operators/pick random[x]to[y] block.	mouse
16	Buying food for a picnic Students will learn how to work with variables: setting different starting values, using conditionals to compare variables' value, changing variables' value, using variables for counting (un)healthy food. In addition, they will repeat adding text, joining texts and if statement.	picnic
17	Operations Students will improve their knowledge on variables, random numbers, loops, broadcast.	operation
18	Recycling Students will improve their knowledge and will extend the game scenario with new objects, code and changing code with respect to new sprites. They will be trained in code refactoring.	recycling
19.1	Play a piano_1 Students will learn about melodies coding and playing and will improve their knowledge about variables, loops, conditional, broadcast and other events.	music

19.2	Play a piano_2 Students will learn how to play music and change costume by clicking on a sprite.	piano
20	Test Students will improve their knowledge and will extend the game scenario with new background, code and changing code with respect to new stages.	test
21	Simplified PACMAN game Students will review their knowledge about movement inside a labyrinth with the use of sense color block. They will learn the concept of cloning the object with position restrictions and how to create a very simple non player character with its own random movement.	pacman

Cenários em Esloveno

N.	TÍTULO	CODE
12	**Lovljenje zdrave hrane** Učenec se nauči naključnega premikanja za X korakov in naključne izbire pozicije ter uporabo spremenljivk in pogojnih stavkov za preprečevanje drugih dogodkov.	lovljenje_hrane_slo
13	**Sestavi zgodbo** Učenec se nauči načrtovanja pripovedovanja zgodb, uporabe pošiljanja obvestil za sinhronizacijo dejavnosti likov ter zamenjave ozadij.	zgodba_slo
14	**Onesnažen zrak** Učenec se nauči risanja, spreminjanja barve, debeline svinčnika ter uporabe spremenljivke in pogojev za izvedbo drugega dogodka.	zrak_slo
15	**Ulovi miš** Učenec se seznani s konceptom dodeljevanja naključnih vrednosti in se nauči, kako uporabiti operator naključno število od_[x]_do_[y].	mis_slo
16	**Kupovanje hrane za piknik**	piknik_slo
17	**Računanje** Učenec izboljša svoje znanje o spremenljivkah, naključnih številkah, zankah ter pošiljanju obvestil.	racunanje_slo
18	**Recikliranje**	recikliranje_slo
19.1	**Zaigraj na klavir 1** Učenec se nauči sestavljanja melodij ter izboljša svoje znanje o spremenljivkah, zankah, pogojih, pošiljanju obvestil ter drugih dogodkov.	ples_slo

19.2	Zaigraj na klavir 2 Učenec se nauči predvajati glasbo in liku spremeniti obleko s klikom nanj.	klavir_slo
20	Test Učenec izboljša svoje znanje in igro razširi z novimi ozadji, doda in spremeni kodo.	test_slo
21	Enostavni Pacman Učenec ponovi s pomočjo zaznavanja barve liku omejiti gibanje, spozna koncept kloniranja z omejitvami gibanja in se nauči ustvarjanja lika, ki se samostojno nakljčno premika po labirintu.	pacman_slo

Cenários em Italiano

N.	TÍTULO	CÓDIGO
12	Raccogliere i cibi salutari Gli studenti impareranno a muoversi casualmente per X passi, a scegliere una posizione, ad usare variabili e condizioni per prevenire altri eventi.	cibo
13	Alice nel Paese delle Meraviglie Gli studenti impareranno a pianificare una storia, ad utilizzare i messaggi inviati e ricevuti per la sincronizzazione delle attività degli Sprite e per i cambiamenti di scena.	alice
14	Disegno Lo studente imparerà a disegnare con Snap!, a cambiare il colore e lo spessore della penna e ad usare le variabili e le condizioni che determinano nuovi eventi.	disegno
15	Alla caccia di un topolino! Gli studenti comprenderanno il concetto di assegnazione di più valori casuali variabili. Impareranno come usare gli Operatori / scegliere il blocco casuale da [x] a [y].	topolino
16	Acquistare cibo per un picnic Gli studenti impareranno a lavorare con le variabili: impostare diversi valori iniziali, usare i condizionali per confrontare il valore delle variabili, cambiare il valore delle variabili, usare le variabili per contare alimenti sani (e non). Inoltre, aggiungeranno il testo, useranno l'unione di testi e la condizione "if".	picnic_1
17	Operazioni Gli studenti miglioreranno le loro conoscenze su variabili, numeri casuali, cicli e broadcast.	operazioni
18	Raccolta differenziata Gli studenti miglioreranno le loro conoscenze per estendere lo	riciclo

	scenario di gioco con nuovi oggetti, codice e cambiando codice rispetto ai nuovi Sprite.	
19.1	Suonare il piano_1 Gli studenti impareranno a codificare le melodie e miglioreranno le loro conoscenze su variabili, loop, condizionale, trasmissione e altri eventi.	music_1
19.2	Suonare il piano_2 Gli studenti impareranno a suonare e a cambiare costume facendo clic su uno Sprite	piano_2
20	Test Gli studenti miglioreranno le loro conoscenze ed estenderanno lo scenario del gioco con nuovi background, codici e sue modifiche.	test_1
21	PACMAN Gli studenti rivedranno le loro conoscenze sul movimento all'interno di un labirinto con l'uso dei blocchi dei sensori del colore. Impareranno il concetto di clonazione dell'oggetto con restrizioni di posizione e a creare un personaggio "non giocatore".	pacman_1

Cenários em Croata

N.	TÍTULO	CÓDIGO
12	**Hvatanje zdrave hrane** Učenici će naučiti kako nasumično pomicati za X koraka i odabrati položaj i također kako koristiti varijable i uvjete za sprječavanje drugih događaja.	hvatanje
13	**Pričam ti priču** Učenici će naučiti kako ispričati priču, kako koristiti poruke i kako promijeniti pozadinu pozornice.	prica
14	**Crtanje** Učenici će naučiti kako se crta pomoću Snap!-a, promijeniti boju i debljinu olovke te kako koristiti varijable i uvjete koji uzrokuju novi događaj.	crtanje
15	**Uhvati miša** Učenik će se upoznati s konceptom dodjeljivanja više varijabli slučajnim vrijednostima. Naučit će kako koristiti blok Operatori / slučajni broj od [x] do [y].	miš
16	**Kupnja hrane za piknik** Učenici će naučiti kako raditi sa varijablama: postavljanje različitih početnih vrijednosti, korištenjem uvjeta za usporedbu vrijednosti varijabli, promjenom vrijednosti varijabli, korištenjem varijabli za brojanje (ne) zdrave hrane. Osim toga, ponovit će dodavanje i spajanje teksta te naredbu ako.	piknik
17	**Operacije** Učenici će poboljšati svoje znanje o varijablama, slučajnim brojevima, petljama, emitiranju.	operacije
18	**Recikliranje** Učenici će poboljšati prethodno stečeno znanje te će proširiti scenarij igre sa novim objektima, kodom i promjenom koda s obzirom na nove	recikliranje

	objekte. Učenici će moći restrukturirati kod.	
19.1	Sviranje klavira_1 Studenti će upoznati kodiranje i sviranje melodija te će poboljšati svoje prethodno stečeno znanje o varijablama, petlji, uvjetnim, emitiranim i ostalim događajima.	glazba
19.2	Sviranje klavira_2 Učenici će naučiti svirati glazbu i mijenjati kostim klikom na sprite.	klavir
20	Test Učenici će poboljšati svoje znanje i proširiti scenarij igre s novom pozadinom, kodom i promjenom koda u odnosu na nove pozornice.	test_cro

Cenários em Búlgaro

N.	TÍTULO	CÓDIGO
12	**Подбор на здравословна храна** Учениците ще се научат как да се придвижат на случаен принцип с X стъпки и да изберат позиция, както и как да използват променливи и условности за предотвратяване на друго събитие.	food_bg
13	**Разказвач** Учениците ще научат как да планират разказването на истории, как да използват излъчвани съобщения за синхронизиране на дейностите на спрайтове и промени на сцената.	story_bg
14	**Рисуване** Ученикът ще се научи да рисува в Snap !, да променя цвета и дебелината на писалката и да използва променливи и условия, които причиняват ново събитие.	drawing_bg
15	**Хванете мишката** Учениците ще бъде запознати с концепцията за присвояване на случвайни стойности на променливи. Ще се научат как да използват блока Operators/pick random[x]to[y] block.	mouse_bg
16	**Купуване на храна за пикник** Учениците ще се научат как да работят с променливи: задаване на различни начални стойности, използване на условия за сравняване на стойностите на променливите, промяна на стойностите на променливите, използване на променливи за броене на (не) здравословна храна. В допълнение, те ще разучат добавяне на текст, съединяване	picnic_bg

	на текстове и if оператор.	
17	Операции Учениците ще подобрят знанията си за променливи, случайни числа, цикли, Broadcast.	oper_bg
18	Рециклиране Учениците ще подобрят знанията си и ще разширят сценария на играта с нови обекти, код и променлив код по отношение на новите спрайтове. Те ще бъдат обучени как да редактират кодове.	recycling_bg
19.1	Посвири на пиано Версия 1 Учениците ще се запознаят с кодирането и свиренето на мелодии и ще подобрят знанията си за променливи, цикли, условия, излъчване и други събития.	music_bg
19.2	Посвири на пиано Версия 2 Учениците ще се научат как да свирят музика и да сменят костюма, като кликнат върху спрайт.	piano_bg
20	Тест Учениците ще подобрят знанията си и ще разширят сценария на играта с нов фон, код и променлив код по отношение на новите сцени.	test_bg
21	Опростена игра PACMAN Учениците ще приложат своите знания за движение в лабиринт с помощта на сензорен цветен блок. Те ще научат концепцията за клониране на обект с ограничения за позицията и как да създадат много прост персонаж, който не е играч, с опция за произволно движение.	pacman_bg

Cenários em Grego

N.	TÍTULO	CÓDIGO
12	**Πιάνοντας υγιεινά τρόφιμα** Οι μαθητές θα μάθουν πώς να κινούνται για X τυχαία βήματα και να επιλέγουν μια θέση και επίσης πώς να χρησιμοποιούν μεταβλητές και συνθήκες για την πρόληψη άλλου συμβάντος.	food_gr

13	Διήγηση ιστορίας Οι μαθητές θα μάθουν πώς να σχεδιάζουν την αφήγηση μιας ιστορίας, πώς να χρησιμοποιούν τα μηνύματα μετάδοσης για συγχρονισμό των δραστηριοτήτων των στοιχείων με τις αλλαγές στη σκηνή.	story_bg
14	Ζωγραφική Ο μαθητής θα μάθει να ζωγραφίζει με το Snap!, να αλλάζει το χρώμα και το πάχος του στυλό και πώς να χρησιμοποιεί μεταβλητές και συνθήκες για ένα νέο συμβάν.	drawing_gr
15	Πιάσε το ποντίκι Ο μαθητής θα μάθει την έννοια της πολλαπλής ανάθεσης τυχαίας τιμής. Θα μάθει πώς να χρησιμοποιεί τους τελεστές / επιλέγει τυχαία [x] έως [y].	mouse_gr
16	Αγοράζοντας φαγητό για πικνίκ Οι μαθητές θα μάθουν πώς να δουλεύουν με μεταβλητές: ανάθεση διαφορετικής αρχικής τιμής, χρήση συνθήκης για σύγκριση της τιμής των μεταβλητών, αλλαγή της τιμής των μεταβλητών, χρήση μεταβλητών για την καταμέτρηση (μη) υγιεινών τροφίμων. Επιπλέον, θα επαναλάβουν την προσθήκη κειμένου, την ένωση κειμένων και τη δήλωση εάν.	picnic_gr
17	Λειτουργίες Οι μαθητές θα βελτιώσουν τις γνώσεις τους σχετικά με μεταβλητές, τυχαίους αριθμούς, βρόχους και μετάδοση.	oper_gr
18	Ανακύκλωση Οι μαθητές θα βελτιώσουν τις γνώσεις τους και θα επεκτείνουν το σενάριο του παιχνιδιού με νέα αντικείμενα, κώδικα και ανακατασκευή κώδικα σε σχέση με τα νέα στοιχεία. Θα εκπαιδευτούν στην ανακατασκευή κώδικα.	recycling_gr

19.1	Παίξτε πιάνο_1 Οι μαθητές θα μάθουν να γράφουν κώδικα και να παίζουν μελωδίες και θα βελτιώσουν τις γνώσεις τους σχετικά με μεταβλητές, βρόχους, συνθήκες, μετάδοση γεγονότων και άλλα γεγονότα.	music_gr
19.2	Παίξτε πιάνο_2 Οι μαθητές θα μάθουν πώς να παίζουν μουσική και να αλλάζουν κοστούμι κάνοντας κλικ σε ένα στοιχείο.	piano_gr
20	Δοκιμαστικό μάθημα - Αλλαγή εμφάνισης και στροφή Μάθετε πώς να αλλάζετε την εμφάνιση του στοιχείου για να κάνετε ένα κινούμενο σχέδιο, αλλάζοντας μεταξύ διαφορετικών τύπων περιστροφών.	test_gr
21	Απλοποιημένο παιχνίδι PACMAN Οι μαθητές θα επανεξετάσουν τις γνώσεις τους σχετικά με την κίνηση μέσα σε ένα λαβύρινθο. Θα μάθουν την έννοια της κλωνοποίησης του αντικειμένου με περιορισμό θέσης και πώς να δημιουργήσουν έναν πολύ απλό χαρακτήρα με τη δική του τυχαία κίνηση.	pacman_gr

Cenários em Português

N.	TÍTULO	CÓDIGO
12	Comendo comida saudável Os alunos aprendem a mover-se aleatoriamente X passos e a escolher uma posição. Aprendem a usar variáveis e instruções condicionais para prevenir um evento.	food_PT
13	Narrativa Os alunos aprendem a planejar a narração de histórias e como usar mensagens para sincronização das atividades dos sprites e mudanças de palco.	storytelling_PT
14	Desenhando Os alunos aprendem a desenhar no Snap!, a alterar a cor e a espessura da caneta e a usar variáveis e condicionais que geram um novo evento.	drawing_PT
15	Apanhar o rato Os alunos aprendem o conceito de atribuição de valores aleatórios de múltiplas variáveis. Aprendem ainda como usar o bloco de operadores de escolha aleatória [x] a [y].	mouse_PT
16	Comprando comida para um piquenique Os alunos aprendem a trabalhar com variáveis: valores iniciais, operadores condicionais para comparar o valor das variáveis, alterar o valor das variáveis, usar variáveis para contar alimentos saudáveis. Além disso, eles vão adicionar e juntar textos.	picnic_PT
17	Operações Os alunos vão melhorar os seus conhecimentos sobre variáveis, números aleatórios e ciclos.	operation_PT
18	Reciclagem	recycling_PT

	Os alunos vão aumentar um cenário do jogo com novos objetos, código e código em relação a novos sprites. Aprendem ainda sobre refatoração de código.	
19.1	Tocar piano_1 Os alunos aprendem sobre codificação e reprodução de melodias e melhoram os seus conhecimentos sobre variáveis, ciclos, instruções condicionais e outros eventos.	music_PT
19.2	Tocar piano_2 Os alunos aprendem a tocar música e a mudar de traje clicando num sprite.	piano_PT
20	Teste Os alunos irão melhorar os seus conhecimentos alargando o cenário do jogo com um novo fundo, código e código mutável em relação a novos palcos.	test_PT
21	Jogo PACMAN simplificado Os alunos irão rever os seus conhecimentos sobre o movimento dentro de um labirinto com o uso do bloco de reconhecimento de cores. Aprendem o conceito de clonar o objeto com restrições de posição e como criar um personagem não-jogador muito simples com o seu próprio movimento aleatório.	pacman_PT

Cenários em Turco

N.	TÍTULO	CÓDIGO
12	Sağlıklı yiyeceği yakalamak	yemek

	Öğrenciler, belirli bir adım atmak için rastgele hareket etmeyi, bir konum seçmeyi ve ayrıca diğer olayları önlemek için değişkenleri ve koşulluları nasıl kullanacaklarını öğreneceklerdir.	
13	Hikaye Anlatma Öğrenciler hikaye anlatımını nasıl planlayacaklarını, karakterin aktiviteleri ile sahne değişikliklerinin senkronizasyonu için yayın mesajlarının nasıl kullanılacağını öğrenecekler	hikaye
14	İklimi İyileştirmek-Çizim Yapma Öğrenci, Snap!'te çizim yapmayı, rengi ve kalem kalınlığını değiştirmeyi ve yeni bir olaya neden olan değişkenler ile koşullu ifadeleri nasıl kullanacağını öğrenecek.	iklim
15	Fareyi yakalamak Öğrencilere çok değişkenli rastgele değer atama kavramı tanıtılacaktır. Operatörleri nasıl kullanacaklarını ve rastgele [x] ila [y] bloğu seçmeyi öğrenecekler.	fare
16	Piknik için yemek almak Öğrenciler değişkenlerle nasıl çalışılacağını öğrenecekler: farklı başlangıç değerleri belirleme, değişkenlerin değerini karşılaştırmak için koşullu ifadeler kullanma, değişkenlerin değerini değiştirme, sağlıklı yiyecekleri saymak (olmayan) için değişkenler kullanma. Ek olarak, metin eklemeyi, metinleri birleştirmeyi ve Eğer (if) ifadesini tekrarlayacaklar.	piknik
17	İşlemler Öğrenciler değişkenler, rastgele sayılar, döngüler, yayın hakkındaki bilgilerini geliştireceklerdir.	işlem

18	**Geri Dönüşüm** Öğrenciler bilgilerini geliştirecek ve oyun senaryosunu yeni karakterlere göre yeni nesneler, kodlar ve değişen kodlarla genişletecekler. Kod yeniden düzenleme konusunda eğitilecekler	dönüşüm
19.1	**Piyano Çalmak** Öğrenciler melodileri kodlama ve çalma hakkında bilgi edinecek ve değişkenler, döngüler, koşullu, yayın ve diğer olaylar hakkındaki bilgilerini geliştireceklerdir.	Piyano1
19.2	**Piyano Çalmak** Öğrenciler bir karaktere tıklayarak müzik çalmayı ve kostümü değiştirmeyi öğrenecekler.	piyano2
20	**Test- Kostümleri değiştirmek ve Dönüşler** Farklı döndürme türleri arasında geçiş yaparak bir animasyon yapmak için karakterin kostümünü nasıl değiştireceğinizi öğrenin	test
21	**Basitleştirilmiş PACMAN oyunu** Öğrenciler duyu renk bloğu kullanarak bir labirent içindeki hareketin nasıl kodlanması gerektiği hakkındaki bilgilerini gözden geçireceklerdir. Nesneyi konum kısıtlamaları ile rastgele hareket edecek şekilde basitçe kodlayacakar ve karakterleri klonlamayı öğrenecekler.	pacman

REFERÊNCIAS

A project of the K-12 Initiative at Stanford University. (2009, 05 06). *Taking Design Thinking to School: Approaches to Integrating the Design Process in K-12 Teaching and Learning.* Retrieved 09 10, 2020, from Stanford Graduate School of Education: https://web.stanford.edu/dept/SUSE/taking-design/presentations/Taking-design-to-school.pdf

Alserri, S., Zin, N., & Wook, T. (2018). Alserri, S. A., Zin, N. A. M., & Wook, T. S. M. T.Gender-based Engagement Model for Serious Games. , , 8(4). . *International Journal on Advanced Science, Engineering and Information Technology, 8*(4), 1350-1357.

Baptista, R., & Vaz de Carvalho, C. (2010). Role Play Gaming and Learning. *Learning Technology, 12*(1).

Combéfis, S., Beresnevičius, G., & Dagiene, V. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. *Olympiads in Informatics, 10*.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K., & Utley, J. (2018). *Design Thinking Bootleg.* Retrieved from https://static1.squarespace.com/static/57c6b79629687fde090a0fdd/t/5b19b2f2aa4a99e99b26b6bb/1528410876119/dschool_bootleg_deck_2018_final_sm+%282%29.pdf

Eurostat. (2020, 09 25). *ICT specialists in employment.* Retrieved 11 12, 2020, from eurostat Statistics Explined: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=ICT_specialists_in_employment#Number_of_ICT_specialists

Feldgen, M., & Clua, O. (2004). Games as a motivation for freshman students to learn programming. *34th ASEE/IEEE Frontiers in Education Conference*, (pp. 1079–1084).

Frankovic, I., Hoic-Bozic , N., & Nacinovic-Prskalo, L. (2018). Serious Games for Learning Programming Concepts. *8th Conference edition The Future of Education.* Florence, Italy.

Garris , R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation and learning. Simulation & Gaming. *An Interdisciplinary Journal of Theory, Practice and Research, 33*(4), 43-56.

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy.* New York: Palgrave/Macmillan.

Gee, J. P. (2007). Are video games good for learning? *Curriculum & Leadership Journal, 5*(1).

Geist, E. (2016). Robots, Programming and Coding, Oh My! *Childhood Education, 92*(4), 298-304. doi:10.1080/00094056.2016.1208008

Georgieva, R., & Tuparova, D. (2019). Educatiomnal Computer Game-Based Environments for Teaching Programming and Developmnt of an Algorithmic Thinking - Comparative AnalysisITHMIC THINKING – COMPARATIVE ANALYSIS. *Mathematics and Education in Mathematics, Forty-eighth Spring Conferenceof the Union of Bulgarian Mathematicians* (pp. 150-156). Union of Bulgarian Mathematicians. Retrieved from http://www.math.bas.bg/smb/2019_PK/tom/pdf/150-156.pdf

Georgieva, R., & Tuparova, D. (2020). Design Thinking - helps in school eucation. *Anniversary International Scientific Conference "Synergeticsand ReflectioninMathematics Education"*, (pp. 341-347). Pamporovo.

Grivokostopoulou, F., Perikos, I., & Hatzilygeroudis, I. (2016). An Educational Game for Teaching Search Algorithms. *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016)* (pp. 129–136). Setubal. Portugal: SCITEPRESS - Science and Technology Publications, Lda.

Gross, B. (2003). The impact of videogames in education. 8(7). *First Monday, 8*(7).

Hijon-Neira, R., Velazquez-Iturbide, A., Pizarro-Romero, C., & Carrico, C. (2015). Serious games for motivating into programming. *Frontiers in Education Confere.*

IDEO Design thinking. (2008, 09 7). *Definitions of design thinking*. Retrieved 11 20, 2020, from https://designthinking.ideo.com/blog/definitions-of-design-thinking

Innovation Starter. (2015, 06 04). *What is design thinking?* Retrieved 11 20, 2020, from http://innovationstarterbox.bg/resources/kakvo-e-dizain-mislene/

Jemmali , C. (2016). May's Journey: A serious game to teach middle and high school girls programming. Worcester Polytechnic Institute. Retrieved from https://digitalcommons.wpi.edu/etd-theses/455

Johnson, C., & all, a. (2016). Game Development for Computer Science Education. *the 2016 ITiCSE Working Group Reports.*

Johnston, R. T., & de Felix, W. (1993). Learning from video games. *Computer in the Schools* , 199-233.

Kafai, Y. (1995). Making game artifacts to facilitate rich and meaningful learning. *Annual Meeting of the American Educational Research Association*, (pp. 1–20). San Francisco.

Karakehayova, M. (2019). Opportunities for Using Design Thinking in School Education. *Education and Development, 3*.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences, 47*, 1991-1999.

Kelleher , C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proc. CHI 2007.*, (pp. 1455-1464).

Luka, I. (2014). Design thinking in pedagogy. 2014, 5,. *The Journal of Education, Culture, and Society, 5*, 63-74.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). CMX: Implementing an MMORPG for learning programming, 8th European Conference on Games Based Learning. *8th European Conference on Games Based Learning - ECGBL 2014.* Berlin.

Malone, T. (1981). What makes computer games fun? *Byte, 6*(12), 258-277.

Malone, T. W. (1981b). Toward a theory of intrinsically motivating instruction. *Cognitive Science, 4*.

Mathrani, A., Christian, S., & Ponder-Sutton, A. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. *Educational Technology & Society, 19*(2), 5-17.

Meerbaum-Salant , O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239-264. doi:10.1080/08993408.2013.832022

Michael, D. R., & Chen, S. (2006). *Serious Games: Games That Educate, Train, and Inform.* Boston: Course Technology PTR.

Miljanovic, M., & Bradbury, J. (2016). Robot on!: a serious game for improving programming comprehension. *Proceedings of the 5th International Workshop on Games and Software Engineering.* Austin, Texas, USA.

Miljanovic, M., & Bradbury, J. (2018). A Review of Serious Games for Programming. *4th Joint International Conference on Serious Games.* Darmstadt, Germany.

Naiman, L. (2019, 06 10). *Design Thinking as a Strategy for Innovation*. Retrieved 11 21, 2020, from Creativity at work: https://www.creativityatwork.com/design-thinking-strategy-for-innovation/

Nančovska Šerbec, I., Kaučič, B., & Rugelj, J. (2008). Pair programming as a modern method of teaching computer science. *International journal on emerging technologies in learning, 3*, 45-49.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. , 191, . *Procedia - Social and Behavioral Sciences, 191*, 1479-1482. doi:doi:10.1016/j.sbspro.2015.04.224

Paavola, S., & Hakkarainen, K. (2005). The Knowledge Creation Metaphor – An Emergent Epistemological Approach to Learning.14. *Sci Educ, 14*, 535-546.

Phan, M., Jardina, J., Hoyle, S., & Chaparro, B. (2012). Examining the role of gender in video game usage, preference, and behavior. *Human Factors and Ergonomics Society 51*, (pp. 1496–1500.).

Pivec, M., & Kearney, P. (2007). Games for Learning and Learning from Games. *Informatica, 31*, 419-423.

Pivec, M., Koubek, A., & Dondi, C. (2004). *Guidelines for Game-Based Learning. Lengerich.* Pabst Science Publishers.

Prensky , M. (2001). *Digital Game-Based Learning.* New York: Mc-Graw-Hill.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond , E., Brennan , K., . . . Kafai, Y. (2009). Scratch: Programming for All. *Communication of the ACM, 52*, 60-67.

Rieber , L. P., Smith , L., & Noah, D. (1998). The value of serious play. *Educational Technology, 38*(6), 29-37.

Rugelj, J. (2016). Serious computer games design for active learning in teacher education. (C. Carvalho , P. Escudeiro, & A. Coelho, Eds.) *Serious games, interaction, and simulation : revised selected papers. Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, 161*, 94-102.

Rugelj, J., & Lapina, M. (2019). Game design based learning of programming. In J. Rugelj, & M. Lapina (Ed.), *Proceedings of the International Scientific Conference Innovative Approaches to the Application of Digital Technologies in Education and Research (SLET-2019), May 20-23, CEUR workshop proceedings. 2494.* Stavropol-Dombay, Russia: Aachen: CEUR-WS.

Shabanah, S., Chen, J., Wechsler, H., Carr, D., & Wegman, E. (2010). Designing Computer Games to Teach Algorithms. *Seventh International Conference on Information Technology: New Generations, ITNG 2010*, (pp. 1119-1126). Las Vegas, NV.

Shute, V. J., & Ke, F. (2012). Games, Learning, and Assessment. In D. Ifenthaler, D. Eseryel, & X. Ge, *Assessment in Game-Based Learning: Foundations, Innovations, and Perspective.* New York, USA: Springer.

Shute, V. J., Rieber, L. P., & Van Eck, R. (2012). Games ... and ... Learning. In R. A. Reiser, & J. V. Dempsey, *Trends and issues in instructional design and technology (3rd ed.).* (pp. 321-332). Boston: Pearson Education.

Sykes, E. (2007). Determining the Effectiveness of the 3D Alice Programming Environment at the Computer Science I Level. *Journal of Educational Computing Research, 36*(2), 223-244. doi:10.2190/J175-Q735-1345-270M

Tuparova, D., Nikolova, E., & Tuparova, E. (2020). Integrated game-based learning in an informatics secondary course: Is there a difference between girls' and boys' achievements? *CSEDU 2020 - Proceedings of the 12th International Conference on Computer Supported Education*, *1*, pp. 702-709. Retrieved 10 12, 2020, from https://www.scitepress.org/Link.aspx?doi=10.5220/0009818407020709

Tuparova, D., Tuparov, G., & Veleva, V. (2019 b). Girls' and boys' viewpoint on educational computer games. *European Conference on Games-based Learning*, (pp. 757–766).

Vahldick, A. (2017). Aperfeiçoamento das Competências de Resolução de Problemas na Aprendizagem Intodutória de Programação de Computadores usando um jogo sério digital. Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Retrieved from http://hdl.handle.net/10316/79528

van Eck, R. (2006). Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless. *EDUCAUSE Review, 41*(2).

Vermeulen, L., Van Looy, J., Courtois, C., & De Grove, F. (2011). Girls will be girls: a study into differences in game design preferences across gender and player types. *Under the mask: perspectives on the gamer conference*, (pp. 1-21).

Weintrop, D., & Wilensky, U. (2015). To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. *Interaction Design and Children*, (pp. 199-208).

Whitton, N. (2009). *Learning with Digital Games: A Practical Guide to Engaging Students in Higher Education.* New York: Routledge.

Whitton, N., & Moseley, A. (2012). *Using Games to Enhance Learning and Teaching: A Beginner's Guide.* New York: Routledge.

Wolz, U., Barnes, T., & Bayliss, J. (2009). Girls do like playing and creating games. *ACM SIGCSE Bulletin, 41*(1), 199–200.

Wolz, U., Maloney, J., & Pulimood, S. (2008). 'Scratch' Your Way to Introductory CS. *SIGCSE Bulletin, 40*, 298-299.

Zapušek, M., & Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Trans. on Game-Based Learning, 13*, 1-12.