temática ISSN 1807-8931

Dungeon-Bot: um jogo de programação de linguagem visual baseado no Portugol

Dungeon-Bot: a visual language programming game based on Portugol

Vitor Eduardo SILVA[1]
Ewerton Eyre de Morais ALONSO[2]

Resumo

A complexidade das linguagens de programação motiva a adoção de estratégias e ferramentas que auxiliem os alunos na aprendizagem desse conteúdo. Uma das soluções utilizada é o uso de jogos de programação, que ajudam o aluno a desenvolver a lógica e o pensamento computacional de maneira lúdica com uma linguagem visual. Diante deste contexto, o objetivo deste artigo é apresentar o Dungeon-Bot, um jogo de programação com linguagem visual e comandos baseados no Portugol. Também é apresentada uma análise e comparação de jogos similares da indústria com o Dungeon-Bot. Foi utilizada a metodologia de estudo de caso para definição, análise e seleção dos jogos a serem comparados. Como resultado, são apresentados testes do jogo realizados com o público-alvo e uma tabela comparativa entre os jogos similares analisados e o Dungeon-Bot. Os resultados permitiram concluir que o jogo, comparado aos demais, se destacou nos conceitos de programação ensinados e no uso do pensamento computacional.

Palavras-chave: Linguagem Visual. Jogos de programação. Portugol. Unity.

Abstract

The complexity of programming languages motivates the adoption of strategies and tools that help students to learn this content. One of the solutions used is programming games, which help the student to develop logic and computational thinking in a playful way with a visual language. Thus, the aim of this article is to present Dungeon-Bot, a programming game with visual language and commands based on Portugol. An analysis and comparison of similar games from the industry with the Dungeon-Bot is also presented. The case study methodology was used to define, analyze and select the games to be compared. As a result, game tests performed with the target audience and a comparative table between the analyzed similar games and the Dungeon-Bot are presented. The results allowed us to conclude that the game, compared to the others, stood out in the programming concepts taught and in the use of computational thinking.

Keywords: Visual language. Programing games. Portugol. Unity.

[1] Graduado em Design de Jogos e Entretenimento Digital pela Universidade do Vale do Itajaí – UNIVALI, Balneário Camboriú/SC. E-mail: vitoredu.ves@gmail.com

[2] Mestre em Ciências da Computação pela Universidade Federal de Santa Catarina – UFSC. Professor do Curso de Design de Jogos e Entretenimento Digital da Universidade do Vale do Itajaí – UNIVALI, Balneário Camboriú/SC. E-mail: ewertonalonso@univali.br

Introdução

A dificuldade encontrada por alunos para aprender programação é um problema comum e gera um alto índice de reprovação e desistência nos cursos da área de tecnologia (PINTO, 2019). Para Gomes e Mendes (2007), o alto nível de abstração exigido pela natureza da programação e a sintaxe complexa utilizada pelas linguagens são alguns dos motivos que contribuem para a falta de motivação dos alunos. Para Gomes e Mendes (2007), os métodos de ensino também são uma causa para a dificuldade na aprendizagem, pois focam principalmente nos detalhes da sintaxe ao invés da lógica, da resolução de problemas e em tornar a aprendizagem mais interessante.

Buscam-se maneiras de ensinar programação que realcem a lógica e a resolução de problemas e que tornem a aprendizagem mais interessante e prazerosa. Aprender programação desde jovem pode trazer diversos benefícios, como o desenvolvimento do pensamento computacional. Wing (2006) define o pensamento computacional como um método eficiente para resolver problemas, desenvolver sistemas e entender o comportamento humano, utilizando conceitos fundamentais da ciência da computação.

Para minimizar as dificuldades encontradas no aprendizado de programação, são usados jogos educacionais, pois estimulam a criatividade, curiosidade, autoconfiança, motivam o aluno e tornam a aprendizagem prazerosa e concreta. (RAPKIEWICZ *et al.*, 2006)

Existem jogos educacionais que incorporam elementos de programação em seu gameplay, ensinam o jogador a programar e desenvolvem o pensamento computacional. Os primeiros jogos desse gênero utilizavam interface texto para o usuário digitar comandos. Com o passar do tempo, jogos com linguagem visual apareceram, permitindo que o usuário programasse por meio da montagem de blocos de comando. Este novo estilo de programação visual acabou se mostrando útil, pois facilitava a aprendizagem à medida que auxiliava na resolução de problemas (SILVA *et al.*, 2015).

Atualmente, é possível encontrar diversos jogos desse gênero, tanto em linguagem visual quanto textual, tendo como objetivo auxiliar no aprendizado de programação. Esses jogos apresentam diferentes estilos, tanto em relação a linguagem utilizada quanto aos elementos de programação abordados em seu gameplay. Também é possível encontrar ferramentas que, embora não sejam consideradas jogos, permitem aos seus usuários criar

suas próprias aplicações utilizando uma linguagem visual, tais como o Scratch (RESNICK *et al.*, 2009) e o App Inventor (WOLBER, 2011).

Porém, a maior parte desses jogos possui interface e textos no idioma inglês que, ao ser incorporado à complexidade já existente da lógica de programação, adiciona um grau de dificuldade a mais na aprendizagem para aqueles que não dominam o idioma, sobretudo para as crianças e adolescentes, que muitas vezes são o principal público-alvo desses jogos.

A busca por desenvolver ferramentas e jogos que auxiliem na aprendizagem de programação e no desenvolvimento do pensamento computacional não é recente, a exemplo de Santos *et al*. (2018), Matos *et al*. (2016), Frose e Silva (2019), Kazimoglu *et al*. (2012), Monclar, Silva e Xexéo (2018) e Miljanovic e Bradbury (2018).

Diante deste contexto, o objetivo deste artigo é apresentar o desenvolvimento do jogo gratuito chamado Dungeon-Bot, para o ensino de conceitos básicos de lógica de programação de forma lúdica e com uma linguagem visual baseada no Portugol, uma pseudolinguagem escrita em português utilizada para introduzir lógica de programação em cursos de tecnologia.

Metodologia

A metodologia aplicada a este artigo foi o estudo de caso (BRANSKI; FRANCO; JÚNIOR, 2010), seguindo as etapas propostas por Branski, Franco e Júnior (2010).

Na etapa de delineamento de pesquisa foi estabelecido o objetivo de buscar e testar jogos de programação visual, investigando e analisando quais conteúdos de programação são abordados e que aspectos didáticos estes jogos utilizam para apresentar estes elementos.

Para a etapa de desenho da pesquisa, foram estabelecidas as métricas a serem utilizadas para analisar e comparar os jogos, sendo elas:

- **Pensamento Computacional**: o jogo possui elementos que incentivam o uso das 4 etapas do pensamento computacional: decomposição, reconhecimento de padrões, abstração e algoritmo (FAÊDA; BAFFA; PEREIRA, 2020).
- **Elementos de programação**: quais são os elementos de programação abordados pelo jogo (laços, variáveis, funções etc.).

- **Linguagem visual**: qual linguagem visual o jogo utiliza (ícones ou blocos de texto).
- **Plataforma**: em quais plataformas o jogo está disponível (PC, Android e/ou IOS).
- **Dicas ou tutoriais**: o jogo fornece dicas ou tutoriais de como utilizar seus recursos caso o jogador sinta dúvida em algum momento.
- **Liberdade de utilização de comandos**: o jogo permite que o jogador utilize quantos comandos achar necessário, permitindo que as fases sejam resolvidas com mais de uma maneira.
- **Utiliza Portugol**: utiliza comandos ou instruções escritos em português.

Para preparação e coleta de dados foram estabelecidos critérios de inclusão (jogos com programação visual, utilizando blocos de texto ou ícones, disponíveis para computador ou dispositivos Android). Como critério de exclusão, foi estabelecido que não seriam incluídos jogos não gratuitos ou aqueles que não estão mais disponíveis em suas respectivas plataformas.

Para pesquisa foram utilizadas as plataformas do Google Scholar e os anais do SBGames cujos resultados foram: LightBot: Code Hour (SPRITEBOX LLC, 2008), RoboZZle (OSTROVSKY, 2009), The Foos (CODESPARK, 2021), Blockly-Games (Labirinto) (GOOGLE, 2021), Code.Org (Labirinto Clássico) (CODE.ORG, 2021) e Cargo-Bot (TWO LIVES LEFT, 2012).

Por fim, foi iniciada a etapa de análise de casos e entre os casos com a elaboração de relatórios, com cada jogo sendo testado e analisado com base nas métricas citadas anteriormente, gerando a Tabela 1 (seção Resultados e discussões).

O ensino de programação por meio de jogos

O primeiro jogo de programação foi Darwin (VYSSOTSKY, 1971), cujo objetivo era escrever um algoritmo em um IBM7090 que pudesse se replicar e destruir os algoritmos adversários.

Após isso, outros jogos do gênero começaram a surgir, como RobotWar (METCALF, 2009) e Color Robot Battle (METCALF, 2009), ambos publicados em 1981. Esses dois jogos possuíam um *gameplay* similar, onde era necessário programar

robôs para lutarem entre si em uma arena virtual (METCALF, 2009), inspirando outros que vieram a surgir posteriormente.

Inicialmente, esses jogos necessitavam que o jogador digitasse os comandos através de uma janela de texto para poder cumprir os objetivos. Com o passar do tempo começaram a surgir jogos que utilizavam uma linguagem visual que permitia ao jogador montar o seu código como se fosse um quebra-cabeça. Esse novo estilo de linguagem acabou se mostrando útil para ensinar conceitos e lógica de programação para crianças, adolescentes e iniciantes na área. Hjorth (2017) concluiu que linguagens visuais reduzem a curva de aprendizado, estimulam a criatividade e evitam emoções negativas em relação a erros complexos.

Hoje é possível encontrar jogos de programação visual tanto para crianças quanto para adultos, como Human Resource Machine (TOMORROW CORPORATION, 2015), 7 Billion Humans (TOMORROW CORPORATION, 2018) e CodeCombat (CODECOMBAT INC, 2021).

A dificuldade encontrada por alunos ao aprender programação não é recente. Muitos estudos tentam compreender quais são as limitações encontradas pelos estudantes e quais são os motivos que os levam a se sentirem descontentes com sua experiência em programação (GOMES; TEDESCO; MELO, 2016).

Huggard (2004) relata que estudos psicológicos sobre o aprendizado de programação vêm sendo feitos há mais de 40 anos, mas parecem ter tido pouco impacto nas aulas de programação, de modo que muitos estudantes ainda têm pouco ou nenhuma confiança em suas habilidades de programação.

Para Gomes *et al*. (2008), aprender a programar é complexo e requer esforço, perseverança e uma abordagem especial em relação a métodos de ensino. Para muitos estudantes, o problema começa na fase inicial da aprendizagem quando é necessário compreender e aplicar conceitos abstratos de programação.

Ainda segundo Gomes *et al.* (2008), existem diversas razões que podem interferir no processo de aprendizagem: os métodos de ensino e de estudo, as habilidades e atitudes dos alunos, a natureza da programação e os aspectos psicológicos.

Souza (2009) explica que a programação é uma disciplina com especificidades que exigem que seu ensino seja diferenciado. Já Souza, Silveira e Parreira (2017) explicam que é difícil atribuir a dificuldade de aprendizagem a uma só causa ou definir

os fatores determinantes que dificultam o processo. Logo, cabe aos interessados o empenho de elaborar meios que facilitem e auxiliem o aprendizado.

Diversos trabalhos buscam maneiras de ensinar programação que engajem o aluno no ambiente de aprendizagem. Souza, Silveira e Parreira (2017) apontam para o uso de ferramentas lúdicas como forma de atrair a atenção dos alunos, para que eles aprendam com mais facilidade e consigam desenvolver suas habilidades. Silva, Medeiros e Aranha (2014) constataram que o uso de jogos é eficaz ao ensinar conceitos de programação.

Mesmo com a existência de ferramentas e jogos que possuem o intuito de facilitar o processo de aprendizagem, muitos alunos ainda encontram dificuldade para programar, o que significa que ainda são necessário estudos para resolver este problema. Aprender a programar traz diversos benefícios, pois estimula o pensamento lógico e o raciocínio matemático, a criatividade e a capacidade de resolver problemas, habilidades que certamente serão necessárias para as gerações futuras (SANTOS; SILVA, 2020).

O pensamento computacional

O pensamento computacional é a capacidade de resolver problemas a partir de conhecimentos e práticas da computação (SILVA; PEREIRA; ODAKURA, 2020). Para Wing (2008), o pensamento computacional é um tipo de pensamento analítico que compartilha características com outras áreas, como a abordagem geral de resolver problemas do pensamento matemático, a concepção e avaliação de sistemas complexos do pensamento de engenharia e o entendimento da mente e do comportamento humano do pensamento científico.

Faêda, Baffa e Pereira (2020) e Silva, Pereira e Odakura (2020) definem os elementos que fundamentam o pensamento computacional da seguinte maneira:

- **Decomposição:** consiste em dividir o problema geral em partes menores que sejam mais fáceis de serem gerenciadas, facilitando a resolução.
- **Reconhecimento de padrões:** identificar características comuns entre as pequenas partes e aproximar problemas que já foram solucionados com os que estão sendo abordados.
- **Abstração:** estabelecida por Wing (2008) como a essência do pensamento computacional, a abstração consiste em focar apenas na parte relevante do problema e deixar de lado o que é menos importante.

- **Algoritmos:** é a construção da solução do problema, onde todo o conhecimento é reunido para formular uma estratégia ou um conjunto de instruções claras e ordenadas com o objetivo de resolver o problema.

Embora o pensamento computacional pareça estar apenas relacionado à programação, ele é útil para qualquer área do conhecimento, sendo uma habilidade necessária para todos, não só cientistas da computação (WING, 2006). Para Faêda, Baffa e Pereira (2020) a importância de estimular o pensamento computacional está em utilizar as habilidades de criação de programas computacionais como metodologia para resolver problemas mais complexos.

Para Madeira (2017), o objetivo não é entender a sintaxe de uma linguagem de programação, mas sim a essência de um programa, como ele deve ser construído e quais estruturas são utilizadas para resolver um problema de forma eficiente. Para o autor, este processo de resolução de problemas gera habilidades especificas que deveriam ser desenvolvidas nas crianças e jovens do século XXI, independente de elas virem a se tornar futuros cientistas da computação.

Portugol

O Portugol, ou português estruturado, é uma pseudolinguagem com aplicabilidade apenas didática (MACHADO, 2017) que foi desenvolvida pelos professores Antônio Carlos Nicolodi e Antônio Manso. Originalmente consistia na tradução da linguagem de programação Pascal para o português (MACHADO, 2017).

Entre as vantagens de utilizar o Portugol com alunos iniciantes estão a simplicidade da linguagem e o fato de os comandos estarem em português, evitando problemas com o idioma inglês (NOSCHANG *et al*., 2014). Segundo Souza (2009), o Portugol permite que o aluno se concentre inicialmente na solução do problema proposto sem ter que dominar a sintaxe de uma linguagem real, mas ao mesmo tempo já apresenta conceitos de programação que posteriormente podem ser traduzidos com mais facilidade para ambientes reais de desenvolvimento.

Atualmente, diversas ferramentas utilizam o Portugol como forma de ensinar conceitos de programação como o Portugol IDE (MANSO; OLIVEIRA; MARQUES, 2009), VisualAlg (SOUZA, 2009), WebPortugol (HOSTINS; RAABE, 2007) e o Portugol Studio (NOSCHANG *et al*., 2014). Conforme Noschang *et al.* (2014), a

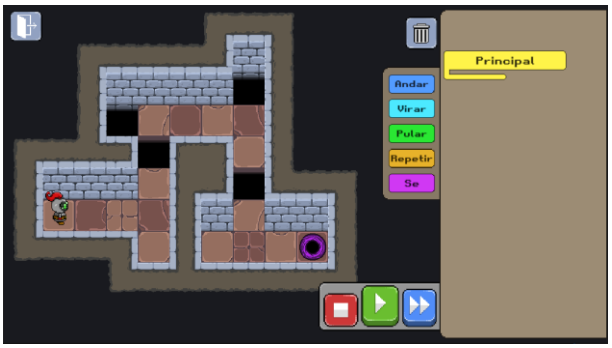
existência dessas ferramentas deixa evidente que existe um interesse significativo por parte de professores e estudantes na utilização do Portugol como linguagem de apoio à aprendizagem de programação.

O projeto Dungeon-Bot

Dungeon-Bot é um jogo digital para ensinar lógica, conceitos básicos de programação e para desenvolver o pensamento computacional, voltado para o público jovem, entre 12 e 15 anos, cuja língua nativa seja o português. O jogo foi desenvolvido utilizando a *engine* Unity 3D e está disponível para computadores e dispositivos móveis. O objetivo do jogo é mover um personagem robô, chamado D-Bot, através de um cenário *top-down* até a saída, representada por um portal roxo. Para isso, o jogador deve construir algoritmos utilizando uma linguagem visual composta por blocos de texto. Estes blocos utilizam uma sintaxe baseada no Portugol.

A interface do jogo, Figura 1, é composta por três áreas: o cenário, onde se encontram o D-Bot, o portal de saída e os obstáculos que o personagem deve superar para atingir o objetivo; os comandos, que representam as ações que o personagem pode realizar; e o código, que é onde o jogador deve posicionar os comandos.

Figura 1: Interface do jogo.



Fonte: O Autor.

Na área do código existe a função “Principal”. Os comandos do jogo são separados em duas categorias:

- Comandos de ação:

Andar, virar, pular, atacar.

- Comandos de programação:

a) **Repetir**: faz o personagem repetir um ou mais comandos por um determinado número de vezes escolhido pelo jogador.

b) **Se**: comando de condição que verifica a presença de objetos a frente, a esquerda, ou a direita do personagem. Caso a condição seja verdadeira, ele executa um ou mais comandos que o jogador escolheu. Para realizar a condição, o comando utiliza os símbolos de "==" (Igual) e "!=" (Diferente), ensinando a sintaxe de comparação encontrada na maior parte das linguagens de programação.

c) **Variável**: em determinado momento do jogo, o personagem encontrará portões de diferentes cores que irão bloquear o caminho até a saída. Para abrir os portões, o D-Bot deve pegar suas respectivas chaves. Porém, para que o personagem consiga coletar as chaves, o jogador deve criar uma variável. As variáveis funcionam da seguinte maneira: primeiro o jogador deve posicioná-las dentro do código, depois ele deve selecionar o tipo de chave que esta variável deve armazenar (Chave Amarela, Chave Verde ou Chave Azul) e, por fim, o jogador deve dar um nome a essa variável. Apenas uma variável de cada cor já é o suficiente para o D-Bot coletar todas as chaves do cenário. Dessa forma, o jogador aprende o conceito de que variáveis possuem tipos, nomes e servem para armazenar coisas.

d) **Função**: o comando "Função" fornece ao jogador a possibilidade de criar funções secundarias com o objetivo de aprender o conceito de reutilização de código. Após executar a função secundaria, o D-Bot volta de onde parou na "Principal".

A área do código possui um sistema de "indentação".

Na parte inferior estão localizados: o botão *play* na cor verde, que serve para iniciar a execução do código; o botão *stop* na cor vermelha, que interrompe a execução do código e reinicia a cena, fazendo todos os objetos voltarem ao estado inicial; e o botão *fast forward,* na cor azul, que faz com que o D-Bot execute os comandos mais rapidamente. Na parte superior existe o botão da lixeira que apaga todos os comandos que estiverem dentro do código. A Figura 2 ilustra comandos do jogo.

Figura 2: Comandos, botões de controle e sistema de "indentação".

Fonte: O Autor.

Os desafios do jogo são representados por fases, divididas em seções que trabalham diferentes conceitos de programação. A versão atual do jogo possui 26 fases separadas nas seguintes seções: comandos básicos (7 fases), loops e condições (9 fases), variáveis (5 fases) e funções (5 fases).

Outro sistema implementado no jogo foi o de missão secundária, cujo propósito foi incentivar o jogador a completar a fase utilizando o menor número de comandos possível. Cada fase possui um número mínimo de comandos (estipulado pelo desenvolvedor) e, caso o jogador consiga completar a fase usando esse número de comandos ou menos, ele ganha uma estrela, que aparece junto da fase no menu de seleção de fases, como mostra a Figura 3.

Figura 3: Seção comandos básicos do menu de seleção de fases.

Fonte: O Autor.

Resultados e discussões

Todos os jogos similares selecionados para análise e comparação apresentam o mesmo estilo de *gameplay* do Dungeon-Bot. São eles:

- LightBot, desenvolvido por Danny Yaroslavski (2014).
- RoboZZle, desenvolvido por OSTROVSKY (2009).
- The Foos, ddesenvolvido pela Code Spark, (GOMES; TEDESCO; MELO, 2016).
- Blockly-Games, um projeto da Google que visa incentivar a formação dos "programadores de amanhã" (RODRIGUES, 2017) que apresenta uma linguagem visual (MARTINS; REIS; MARQUES, 2016).
- Code.Org (MARTINS; REIS; MARQUES, 2016).
- Cargo-Bot, TWO LIVES LEFT (2012).

Incialmente, planejava-se testar o jogo em salas de aula com vários alunos ao mesmo tempo, porém, isso acabou não sendo possível devido a COVID-19. Além disso, pelo fato de as versões mais antigas não possuírem tutoriais dentro do jogo, tornou-se difícil realizar testes remotos, pois o público mais jovem, muitas vezes precisa de instruções dadas pessoalmente para poder entender o funcionamento do jogo. Os testes das primeiras versões do jogo ficaram restritos à um jogador do público-alvo específico com idade de 12 anos.

Os testes começaram na etapa de desenvolvimento e foram extremamente úteis para o aprimoramento do projeto, pois permitiram avaliar a experiência do jogador e seu entendimento do funcionamento dos comandos. Os testes ajudaram a melhorar a *interface*, o *design* das fases e a linguagem visual, que se mostrou eficiente para a representação e organização do código. A utilização do Portugol permitiu que todos os participantes entendessem rapidamente o funcionamento de cada comando, com as únicas exceções sendo o comando “Variável” e “Função”, onde alguns participantes ficaram confusos por não entender o motivo de ter que utilizar esses comandos ou qual o significado desses nomes.

As 4 etapas do pensamento computacional foram observada durante os testes. Os jogadores utilizaram a decomposição para separar as fases em partes menores e entender o problema com mais facilidade. O reconhecimento de padrões foi aplicado para montar

códigos mais otimizados. A abstração foi usada pelos jogadores quando era necessário raciocinar o passo-a-passo que o personagem deveria executar para poder cumprir seu objetivo antes mesmo do código ser construído. Na etapa do algoritmo os jogadores chegaram à solução do problema montando o código usando a linguagem visual.

A Tabela 1 contribuiu para identificar os pontos fortes e fracos do Dungeon-Bot.

Tabela 1: Comparação entre jogos da indústria e o Dugeon-Bot.

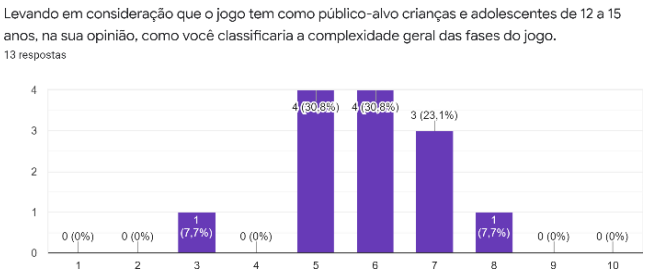
Jogos	Pensamento Computacional	Elementos de programação	Linguagem visual	Plataforma	Dicas e/ou tutoriais	Liberdade de utilização de comandos	Utiliza Portugol
Dungeon-Bot	Decomposição Padrões Abstração Algoritmos	Controle de fluxo Laços Condições Variáveis Funções	Blocos de texto	PC	Tutoriais	Sim	Sim
LightBot: Code Hour	Decomposição Padrões Algoritmos	Controle de fluxo Funções Recursão	Ícones	Android, IOS	Tutoriais	Não	Sim
RoboZZle	Decomposição Padrões Algoritmos	Controle de fluxo Funções Recursão Condições	Ícones	Android, IOS, PC	Tutoriais	Não	Não
The Foos	Padrões Algoritmos	Controle de fluxo Laços	Ícones	Android, IOS, PC	Tutoriais	Sim	Sim
Blockly-Games (Labirinto)	Decomposição Padrões Algoritmos	Controle de fluxo Laços Condições	Blocos de texto	PC	Tutoriais	Não	Sim
Code.Org (Labirinto Clássico)	Decomposição Padrões Algoritmos	Controle de fluxo Laços Condições	Blocos de texto	PC	Dicas e tutoriais	Sim	Sim
Cargo-Bot	Decomposição Padrões Abstração Algoritmos	Controle de fluxo Funções Recursão	Ícones	PC, IOS	Dicas e tutoriais	Sim	Não

Fonte: O Autor.

Com a implementação dos tutoriais no jogo, novos testes foram realizados por 13 desenvolvedores com idades fora do público-alvo. Os *feedbacks* se deram por meio de questionários com perguntas descritivas para verificar se o jogador havia entendido como utilizar cada comando do jogo, se os tutoriais apresentados foram uteis para compreender como o jogo funcionava e se algum erro foi encontrado enquanto jogavam. As respostas das questões objetivas geraram gráficos apresentados abaixo.

O gráfico da Figura 4 indica que o jogo possui dificuldade média, um dos resultados esperados.

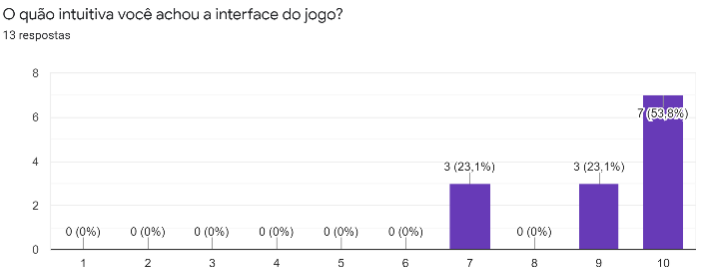
Figura 4: Respostas em relação a dificuldade do jogo



Fonte: O Autor

O gráfico da Figura 5 indica que a interface é intuitiva.

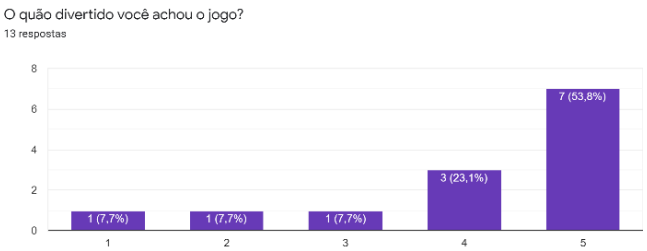
Figura 5: Respostas em relação a dificuldade do jogo.



Fonte: O Autor.

O gráfico da Figura 6 apresenta resultados mistos, indicando que o jogo está divertido e que pode satisfazer um público mais experiente.

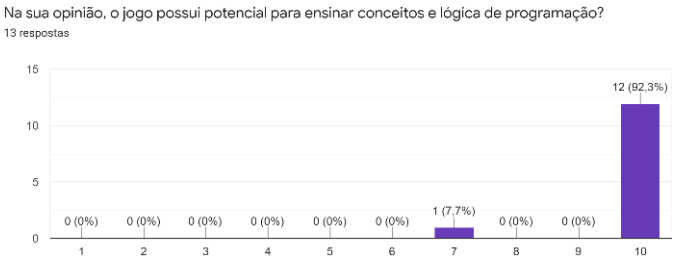
Figura 6: Respostas em relação a dificuldade do jogo.



Fonte: O Autor.

O gráfico da Figura 7 indica que o jogo possui potencial como ferramenta educacional.

Figura 7: Respostas em relação ao potencial educacional do jogo



Fonte: O Autor.

Assim, foi possível concluir que o Dungeon-Bot se diferencia de outros jogos, pois, além de desenvolver o pensamento computacional, consegue ensinar mais conceitos de programação, com destaque para as variáveis que não são abordadas por outros jogos analisados.

Considerações finais

Muitos estudantes encontram desafios na aprendizagem de programação, incentivando a busca por ferramentas que auxiliem no ensino de lógica e pensamento computacional e que possam ser introduzidas logo cedo no ambiente escolar como, por exemplo, os jogos digitais que motivam o aluno a aprender de forma lúdica.

O objetivo deste artigo foi apresentar o Dungeon-Bot, um jogo digital para o ensino de lógica, pensamento computacional e conceitos básicos de programação para jovens por meio de uma linguagem visual inspirada no Portugol. Ao final, foram apresentados os resultados obtidos por meio de testes realizados, além de uma comparação com outros jogos similares do mercado.

Não foi possível avaliar a autonomia do jogo, promovida por meio dos tutoriais implementados e que torna possível ao jogador aprender sem auxílio externo. Espera-se em trabalhos futuros realizar uma avaliação da usabilidade do Dungeon-Bot em ambientes escolares, examinando as interações dos alunos e avaliando se este possui as qualidades necessárias para uma ferramenta educacional.

Referências

BRANSKI, Regina Meyer; FRANCO, Raul A.; JÚNIOR, Orlando Lima. **Métodologia de estudo de casos aplicada à logística**. Regina Meyer Branski Raul Arellano Caldeira Franco Orlando Fontes Lima Jr., p.1–12, 2010.

CODECOMBAT INC. **CodeCombat**. 2021. Disponível em: https://br.codecombat.com/. Acesso em: 17 mai 2021.

CODESPARK. **CodeSpark Academy: Coding App for Kids**. 2021. Disponível em: https://codespark.com/. Acesso em: 22 mai 2021.

CODE.ORG. **Code.org - Learn Computer Science**. 2021. Disponível em: https://studio.code.org/courses. Acesso em: 22 mai 2021.

FAÊDA, Leonardo M; BAFFA, Matheus F O; PEREIRA, Julie s. AI(3P)A: **Uma Metodologia para o Ensino de Lógica de Programação Utilizando Jogos Eletrônicos.** In: SIMPÓSIO BRASILEIRO DE GAMES E ENTRETENIMENTO DIGITAL, 19., 2020, Recife**.** Proceedings of SBGames 2020. **Recife**: Sbc, 2020. p. 537-543. Disponível em: https://www.sbgames.org/proceedings2020/EducacaoFull/209584.pdf. Acesso em: 20 mai 2021.

FROSI, Felipe Oviedo; SILVA, Isabel Cristina Siqueira da. **CodeBots Ensino Lúdico de Conceitos Introdutórios de Programação para Estudantes da Educação Básica**. 18. ed. Rio de Janeiro: Sbc, 2019. 10p. Disponível em: https://www.sbgames.org/sbgames2019/files/papers/EducacaoFull/197906.pdf. Acesso em: 24 mai 2021.

GOMES. Anabela; MENDES. Antonio Jose. **Learning to program - difficulties and solutions**. Academic Conference Paper. Maio, 2007. Disponível em: https://www.researchgate.net/publication/228328491_Learning_to_program_-_difficulties_and_solutions. Acesso em: 02 abr 2021.

GOMES, Anabela *et al*. **Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte.** Revista Portuguesa de Pedagogia, p. 161–179, 2008. DOI 10.14195/1647-8614_42-2_9. Disponível em: https://impactum-journals.uc.pt/rppedagogia/article/view/1647-8614_42-2_9. Acesso em: 18 mai. 2021.

GOMES, Tancicleide C S; TEDESCO, Patricia C de A. R.; MELO, Jeane C B de. **Jogos no Design de Experiências de Aprendizagem de Programação Engajadoras**. V Jornada de Atualização em Informática na Educação, p.39–77, 2016.

GOOGLE. **Blockly Games**. 2021. Disponível em: https://blockly.games/maze?lang=pt-br. Acesso em: 22 mai 2021.

HJORTH. Maria. **Strengths and weaknesses of a visual programming language in a learning context with children**. [*S.l.*: *s.n.*], 2017.

HOSTINS, Higor; RAABE, André. **AUXILIANDO A APRENDIZAGEM DE ALGORITMOS COM A FERRAMENTA WEBPORTUGOL.** In: CONGRESSO

DA SBC, 27., 2007, Rio de Janeiro. **Anais [...].** Itajaí: Sbc, 2007. p.96-105. Disponível em: https://www.researchgate.net/publication/268344218. Acesso em: 23 mai 2021.

HUGGARD. Meriel. **Programming trauma: Can it be avoided**. Proceedings of the BCS Grand Challenges in Computing: Education, January, p.50–51, 2004.

KAZIMOGLU, Cagin *et al*. **A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming.** Procedia - Social And Behavioral Sciences**,** [S.L.], v.47, p.1991-1999, 2012. Elsevier BV. http://dx.doi.org/10.1016/j.sbspro.2012.06.938.

MACHADO. Caìque da Silva. **Tradutor Didático: Um Software Que Auxilia Na Tradução Da Pseudo Linguagem Portugol Para A Linguagem C**. 2017. 65 f. TCC (Graduação) - Curso de Tecnologia em Sistemas de Computação, Universidade Federal Fluminense, Niterói, 2017. Disponível em: https://app.uff.br/riuff/bitstream/1/12825/1/TCC_CAIQUE_DA_SILVA_MACHADO.pdf. Acesso em: 23 mai 2021.

MADEIRA, Charles. **Introdução ao Pensamento Computacional com Scratch.** In: CONGRESSO SOBRE TECNOLOGIAS NA EDUCAÇÃO, 2, 2017, Natal. Paraíba: Ctrl+E, 2017. p.725-730.

MANSO, Antonio; OLIVEIRA, Luis M L; MARQUES, Célio Gonçalo. **Ambiente de Aprendizagem de Algoritmos – Portugol IDE**. [*S.l.*: *s.n.*], 2009. Disponível em: https://www.researchgate.net/publication/277327305. Acesso em: 25 mai 2021.

MARTINS, Ricartty; REIS, Ronaldo; MARQUES, Anna Beatriz. **Inserção da programação no ensino fundamental Uma análise do jogo Labirinto Clássico da Code.org através de um modelo de avaliação de jogos educacionais.** Anais do XXII Workshop de Informática na Escola (WIE 2016), vol.1, Cbie, p.121, 2016. https://doi.org/10.5753/cbie.wie.2016.121.

MATOS, Marilyn Aparecida Errobidarte de *et al*. **Ensinando programação para crianças: um jogo.** In: SIMPÓSIO BRASILEIRO DE GAMES E ENTRETENIMENTO DIGITAL, 15, 2016, São Paulo. **Proceedings of SBGames 2016.** Campo Grande: Sbc, 2016. p.1210-1213.

METCALF, John. **A History of Programming Games 1961-1989**. 2009. Disponível em: http://www.retroprogramming.com/2009/09/history-of-programming-games-1961-1989.html. Acesso em: 17 mai 2021.

MILJANOVIC, Michael A.; BRADBURY, Jeremy S. **A review of serious games for programming.** Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11243 LNCS, no. September, p.204–216, 2018. https://doi.org/10.1007/978-3-030-02762-9_21.

MONCLAR, Rafael Studart; SILVA, Marcelo Arêas; XEXÉO, Geraldo. **Jogos com Propósito para o Ensino de Programação.** In: SIMPÓSIO BRASILEIRO DE GAMES E ENTRETENIMENTO DIGITAL, 17, 2018, Foz do Iguaçu. **Proceedings of SBGames 2018.** Foz do Iguaçu: Sbc, 2018. p.1132-1140. Disponível em: https://www.sbgames.org/sbgames2018/files/papers/EducacaoFull/188132.pdf. Acesso em: 10 abr 2021.

NOSCHANG, Luiz F *et al*. **Portugol Studio: Uma IDE para Iniciantes em Programação**. Anais do XXII Workshop sobre Educação em Computação, p.535–545, 2014.

OSTROVSKY. Igor. **RoboZZle**. 2009. Disponível em: http://robozzle.com. Acesso em: 22 mai 2021.

PINTO. Caroline Siervo. **O ensino da lógica de programação utilizando a ferramenta de aprendizagem Visualg**. 2019. 50f. TCC (Graduação) - Curso de Especialista em Informática Instrumental, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019.

RAPKIEWICZ, Clevi Elena *et al*. **Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais.** Renote**,** [S.L.], v.4, n.2, p.1-11, 22 dez. 2006. Universidade Federal do Rio Grande do Sul. http://dx.doi.org/10.22456/1679-1916.14284.

RESNICK, Mitchel *et al*. **Scratch.** Communications Of The Acm**,** [S.L.], v.52, n.11, p.60-67, nov. 2009. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/1592761.1592779. Disponível em: http://scratch.mit.edu. Acesso em: 3 abr 2021.

RODRIGUES. Rivanilson da Silva. **Um estudo sobre os efeitos do Pensamento Computacional na educação**. 2017. 113f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal de Campina Grande, Campina Grande, 2017. Disponível em: http://dspace.sti.ufcg.edu.br:8080/jspui/bitstream/riufcg/705/1/RIVANILSON%20DA%20SILVA%20RODRIGUES%20-%20DISSERTA%c3%87%c3%83O%20%28PPGCC%29%202017.pdf. Acesso em: 24 mai 2021.

SANTOS, Ian Macedo Maiwald *et al*. **As aventuras espaciais de Cody:** protótipo de jogo para auxiliar no ensino de lógica de programação. 17. ed. Foz do Iguaçu: Sbc, 2018. 4p. Disponível em: https://www.sbgames.org/sbgames2018/files/papers/EducacaoShort/188395.pdf. Acesso em: 24 mai 2021.

SANTOS. Vinicius George dos; SILVA. Soraia Lúcia da. **Educação tecnológica:** o ensino da programação para crianças do Ensino Fundamental através do ambiente Code.Org. Conecte-Se! Revista Interdisciplinar de Extensão, Belo Horizonte, v.4, n.7, p.23-39, mai 2020.

SILVA, Kennedy dos Santos; PEREIRA, Nícolas Pierim; ODAKURA, Valguima Victoria Viana Aguiar. **Proposta de ferramenta educacional para o desenvolvimento do pensamento computacional.** Tecnologias, Sociedade e Conhecimento**,** [S.L.], v. 7, n.1, p.48-70, 4 ago. 2020. Universidade Estadual de Campinas. http://dx.doi.org/10.20396/tsc.v7i1.14706. Disponível em: https://econtents.bc.unicamp.br/inpec/index.php/tsc/article/view/14706/9695. Acesso em: 22 mai 2021.

SILVA. Thiago Reis da; MEDEIROS. Taina Jesus; ARANHA. Eduardo Henrique da S. **Jogos digitais para ensino e aprendizagem de programação: uma revisão sistemática**

da literatura. Anais do XXV Simpósio Brasileiro de Informática na Educação (SBIE 2014), vol. 1, no. October 2014, p.692, 2014.

SILVA, Thiago Reis da *et al*. **Ensino-aprendizagem de programação:** uma revisão sistemática da literatura. Revista Brasileira de Informática na Educação**,** [S.L.], v.23, n.01, p.182-196, 30 abr. 2015. Sociedade Brasileira de Computacao - SB. http://dx.doi.org/10.5753/rbie.2015.23.01.182.

SOUZA, Claúdio Morgado De. VisuaAlg – ferramenta de apoio ao ensino de programação. **Revista TECCEN**, setembro de 2009, vol.2, n.2. Disponível em <http://editora.universidadedevassouras.edu.br/index.php/TECCEN/article/view/234>. Acesso em: 18 mai 2021.

SOUZA. Naidú Gasparetto de; SILVEIRA. Sidnei Renato; PARREIRA. Fábio José. **Proposta de uma metodologia para apoiar os processos de ensino e de aprendizagem de lógica de programação na modalidade de educação a distância**. 2017. 29 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Santa Maria, Santa Maria, 2017. Disponível em: https://repositorio.ufsm.br/handle/1/12842. Acesso em: 14 mai 2021.

SPRITEBOX LLC. **LightBot: Code Hour**. 2008. Disponível em: https://play.google.com/store/apps/details?id=com.lightbot.lightbothoc&hl=pt_BR&gl=US. Acesso em: 22 mai 2021.

TOMORROW CORPORATION. **Human Resource Machine**. 2015. Disponível em: https://store.steampowered.com/app/375820/Human_Resource_Machine/. Acesso em: 17 mai 2021.

TOMORROW CORPORATION. **7 Billion Humans**. 2018. Disponível em: https://store.steampowered.com/app/792100/7_Billion_Humans/. Acesso em: 17 mai 2021.

TWO LIVES LEFT. **Cargo-Bot**. 2012. Disponível em: https://apps.apple.com/br/app/cargo-bot/id519690804. Acesso em: 22 mai 2021.

VYSSOTSKY, Victor A. **Darwin, a game of survivalofthe fittest among programs**. [*S.l.*: *s.n.*], 1971. Disponível em: https://www.cs.dartmouth.edu/~doug/darwin.pdf. Acesso em: 3 abr 2021.

WING, Jeannette M.. **Computational thinking**. 2006. Disponível em: https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf. Acesso em: 02 abr 2021.

WOLBER. David. **App inventor and real-world motivation**. [*S.l.*: *s.n.*], 2011. Disponível em: http://www.acm.org/class/1998. Acesso em: 3 abr 2021.

WING, Jeannette M. **Computational thinking and thinking about computing.** Philosophical Transactions Of The Royal Society A**:** Mathematical, Physical and Engineering Sciences, [S.L.], v.366, n.1881, p.3717-3725, 31 jul. 2008. The Royal Society. http://dx.doi.org/10.1098/rsta.2008.0118.

YAROSLAVSKI, Danny. **How does Lightbot teach programming?** [*S.l.*: *s.n.*], 14 Jul. 2014.