UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

UNIDADE ACADÊMICA DE SERRA TALHADA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOÃO EMERSON NICULAU DA SILVA

PaintCode: um ambiente de programação baseada em blocos para o ensino de conceitos básicos de Programação

Serra Talhada,
Março/2021

João Emerson Niculau da Silva

PaintCode: um ambiente de programação baseada em blocos para o ensino de conceitos básicos de Programação

Trabalho de Conclusão de Curso apresentada ao Curso de Bacharelado em Sistemas de Informação da Unidade Acadêmica de Serra Talhada da Universidade Federal Rural de Pernambuco como requisito parcial à obtenção do grau de Bacharel.

Orientador: Prof. Dr. Richarlyson Alves D'Emery

Serra Talhada,
Março/2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586p Silva, João Emerson Niculau
PaintCode: um ambiente de programação baseada em blocos para o ensino de conceitos básicos de Programação / João Emerson Niculau Silva. - 2021.
32 f. : il.

Orientador: Richarlyson Alves DEmery.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Serra Talhada, 2021.

1. Software Educacional. 2. Programação em Blocos. 3. Pensamento Computacional. 4. Avaliação de Usabilidade. I. DEmery, Richarlyson Alves, orient. II. Título

CDD 004

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE SERRA TALHADA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOÃO EMERSON NICULAU DA SILVA

PaintCode: um ambiente de programação baseada em blocos para o ensino de conceitos básicos de Programação

Trabalho de Conclusão de Curso julgado adequado para obtenção do título de Bacharel em Sistemas de Informação, defendida e aprovada por unanimidade em 04/03/2021 pela banca examinadora.

Banca Examinadora:

Prof. Dr. Richarlyson Alves D'Emery
Orientador
Universidade Federal Rural de Pernambuco

Prof. M.e Glauber Magalhaes Pires
Universidade Federal Rural de Pernambuco

Prof. M.e Heldon José Oliveira Albuquerque
Universidade Federal Rural de Pernambuco

AGRADECIMENTOS

Em primeiro lugar, a Deus, que fez com que meus objetivos fossem alcançados, durante todos os meus anos de estudos. Aos meus pais e minha irmã por todo suporte e apoio que precisei. Aos meus amigos, que sempre estiveram ao meu lado, pela amizade e pelo apoio que me deram. Aos professores, por todos os conselhos, pela ajuda e pela paciência com a qual guiaram o meu aprendizado, principalmente ao professor Richarlyson D'Emery por ter sido meu orientador e ter me guiado durante a realização deste trabalho.

“Se o conhecimento pode criar problemas, não é através da ignorância que podemos solucioná-los”.

(Isaac Asimov)

SUMÁRIO

PaintCode: um ambiente de programação baseada em blocos para o ensino de conceitos básicos de Programação

João Emerson Niculau da Silva[1], Richarlyson Alves D'Emery[1]

[1]Unidade Acadêmica de Serra Talhada (UAST) – Universidade Federal do Rural de Pernambuco (UFRPE)
Av. Gregório Ferraz Nogueira, S/N –56909-535– Serra Talhada – PE – Brasil

{joaoniculau, ricodemery}@gmail.com

Abstract. Introduction: *Programming is a challenge in Computing, both for its complexity and for being a subject not taught in basic education. With educational software, its introduction and practice can be aided.* ***Goals****: To present the PaintCode software that helps teaching and learning of basic programming concepts.* ***Methods:*** *Use Python, the MVC architectural pattern, good design principles and block programming. Usability assessment uses System Usability Scale (SUS) and statistical inference by Chi-Square and ANOVA.* ***Results:*** *SUS is 91.38 points, with best imaginable adjectives SUS and hypothesis is proven a significant contribution to the teaching and learning process of basic programming concepts.*

Resumo. Introdução*: Programação é desafio na Computação, tanto pela complexidade, como por ser assunto não ensinado na educação básica. Com software educacional pode-se auxiliar a sua introdução e prática.* ***Objetivo:*** *Apresentar o software PaintCode que auxilia o ensino e a aprendizagem de conceitos básicos de Programação.* ***Métodos:*** *Utiliza codificação Python, padrão arquitetural MVC, boas práticas de design e programação em blocos. A avaliação de usabilidade é dada por System Usability Scale (SUS) e inferência estatística por Qui-Quadrado e ANOVA.* ***Resultados:*** *Avaliação SUS é de 91,38 pontos, classificado como melhor imaginável e hipótese comprovada da contribuição significante para o processo de ensino e aprendizagem de conceitos básicos de Programação.*

1. Introdução

O ensino de conceitos básicos de Programação é um grande desafio para cursos de Computação, devido ao seu grau de dificuldade, um dos fatores que incitam nesta dificuldade é o fato de não serem abordados assuntos que envolvam pensamento computacional na educação básica na maioria das escolas. O ensino de Pensamento Computacional, além de prover uma base para futuros estudantes de Computação, também pode trazer vários benefícios para que desenvolvam uma melhor compreensão de raciocínio lógico, útil em diversas situações como, por exemplo, ações do dia-a-dia envolvendo Matemática e Física [Pereira e Rapkiewicz 2004].

Dados apresentam que a demanda de profissionais de Computação é maior que a oferta [Brasscom 2020], porém o ensino da Computação na educação básica ainda não faz parte da realidade do Brasil e em boa parte do mundo. Para um país, é extremamente estratégico formar bons profissionais em Computação, pois é um ramo que está presente em todas as atividades econômicas e produtivas [Bezerra e Dias 2014].

Dados de censos dos últimos anos realizados pelo Instituto Nacional de Estudos e Pesquisa Educacionais Anísio Texeira (INEP) [Brasil 2020] para a educação superior nas áreas de Computação e Tecnologias apontam uma grande evasão e que auxiliada ao déficit de pessoas nessas áreas aponta-se a relevância da realização de pesquisas destinadas a criação de ferramentas capazes de dar o suporte inicial para compreensão de conceitos básicos para os ingressantes na área, contribuindo para minimizar a evasão.

Software educacional é um exemplo de ferramenta que pode tratar assuntos de difícil compreensão de forma lúdica e interativa. [Haguenauer et al. 2007] afirmam que softwares lúdicos aplicados a Educação podem ser ferramentas eficientes no processo de ensino e aprendizagem, pois eles motivam o aprendizado, trazendo o assunto de forma divertida, contribuindo também no entendimento do que foi ensinado. Além disso, é possível contextualizar o conhecimento através de desafios em que associam o conteúdo a algum tipo de situação imaginável ou cotidiana. Portanto, a utilização dessas ferramentas como abordagem de aprendizagem dinâmica se mostra capaz de estimular a curiosidade, trazer novas habilidades e conhecimentos.

Sendo assim, este trabalho tem como objetivo geral apresentar um software educacional que auxilia o ensino e a aprendizagem de conceitos básicos de Programação, comumente usados em pensamento computacional, através de um ambiente que explora a programação em blocos. Para o desenvolvimento do mesmo tem-se objetivos específicos: (i) revisar ferramentas e métodos de ensino e aprendizagem de Lógica de Programação; (ii) elaborar os desafios e métodos do software educacional; (iii) modelar o software educacional; (iv) codificar o software educacional; e (v) avaliar a usabilidade do software quanto sua contribuição para o processo de ensino e aprendizagem de Lógica de Programação.

Este artigo está organizado da seguinte forma: na Seção 2 é apresentado o referencial teórico sobre ensino de lógica de Programação e usabilidade de software. Posteriormente, na Seção 3, é apresentado o método, em que é descrito o desenvolvimento do software, suas regras e sua avaliação de usabilidade. Na Seção 4, são apresentados os resultados. Por fim, na Seção 5 é apresentada a conclusão, seguidas das referências que guiaram a pesquisa.

2. Referencial Teórico

2.1. Ensino de Lógica de Programação

[Rocha et al. 2015] afirmam que lógica de Programação é requisito fundamental nos cursos de Computação, pois é um instrumento importante na estruturação de raciocínio lógico e formulação de algoritmos corretos inerentes ao pensamento computacional. Porém, a condução desse estudo é vista com preocupação por diversos alunos, devido às características lógico-matemática, tal que o aluno se sente incapaz de realizar tais abstrações lógicas, dado o enraizado estigma relacionado a esse conteúdo. Uma provável solução apontada por várias pesquisas é a da iniciação do aluno em lógica de Programação ainda no nível básico.

Portanto, diversas abordagens de ensino vêm surgindo atualmente, como é o caso de plataformas que adotam uma estratégia composta por traços lúdicos, ambientação amigável e programação baseada em blocos. [Cavalcante, Costa e Araújo 2016] destacam que ferramentas que contribuem no ensino de programação com base em

blocos são tidas como potenciais estratégias, seja pela proposta interativa e desafiadora ou pela possibilidade de trabalhar com o lúdico. Podendo assim, dessa perspectiva, trazer conceitos básicos da Ciência da Computação, principalmente da Programação, para o ambiente escolar através de desafios de programação de forma divertida, possibilitando a resolução de situações abstratas, de maneira menos complexas e de fácil compreensão.

Com isso, diversas plataformas para o ensino de lógica de programação vêm surgindo como, por exemplo, a utilização de ambientes como Scratch [Oliveira, 2014] e App Inventor [Ramos et al. 2015], consideradas ferramentas de propósito geral, entre outras destacadas como trabalhos relacionados, as quais são mais direcionadas ao objeto de estudo desta pesquisa.

Todavia, softwares educacionais devem ser avaliados antes de serem introduzidos no ambiente acadêmico. [Reategui 2007] enfatiza a importância da usabilidade para atingir a qualidade das experiências ao se fazer uso desses softwares. O autor afirma que os problemas na usabilidade podem não apenas prejudicar o uso do software, mas também prejudicar o ensino dos conteúdos abordados. Sendo assim, na próxima seção trata da Usabilidade de Software.

2.2. Usabilidade de Software

Segundo [Alves e Pires 2002], usabilidade significa a forma como os utilizadores realizam determinadas tarefas eficientemente, efetivamente e satisfatoriamente. Pode-se afirmar que a usabilidade deve ser preocupação constante durante todo o processo de desenvolvimento de um software, visto que é capaz de aumentar os níveis de utilização do software por garantir sistemas mais fáceis de usar e mais eficientes, assegurando uma maior satisfação na sua utilização.

[Silva 2003] afirma que a usabilidade de um software é uma medida de como é fácil e prático usá-lo, ou a probabilidade de que o usuário do sistema não vá encontrar problemas com a interface durante certo tempo de uso em determinadas condições.

Cada avaliação de usabilidade aborda objetivos, necessidades, forma de aplicação, desde princípios puramente de aparência e interação, como as que se preocupam com aspectos pedagógicos. Portanto, se faz necessário usar técnicas e métodos que viabilizem essa avaliação através da seleção de referências que atenda ao propósito da avaliação. Sendo assim, na próxima seção, apresentam-se trabalhos relacionados destinados a auxiliar o processo de ensino e aprendizagem de pensamento computacional bem como do processo de avaliação desses.

2.3. Trabalhos Relacionados

Para a pesquisa de trabalhos presentes da literatura, utilizou-se da ferramenta Google Scholar[1] através de Strings de busca que utilizavam palavras-chaves, como, por exemplo, software educacional e seus subtipos, avaliação, pensamento computacional, lógica de programação e programação baseada em blocos. Os trabalhos que contemplavam o assunto objeto desta pesquisa ou que abordavam o processo de avaliação de um software em que pudessem ser aproveitados nesta pesquisa foram selecionados. Apesar das diversas seleções, nesta seção apresentam-se aqueles em que os pesquisadores consideraram mais relevantes.

[1] https://scholar.google.com.br/

No trabalho intitulado “"Aperta o Play!" Análise da Interação Exploratória em um Jogo Baseado em Pensamento Computacional”, [Falcão e Barbosa 2005] realizaram uma avaliação do jogo Lightbot (Figura 1), um jogo que “tem o intuito de apresentar ao jogador diferentes desafios de Lógica de Programação com um objetivo simples: movimentar um robô (utilizando os ícones de comandos) até o ladrilho azul e acender a lâmpada ali presente, deixando assim o ladrilho amarelo”. O objetivo desse trabalho foi de contribuir para a difusão no ensino básico de jogos digitais educacionais com foco no desenvolvimento de pensamento computacional.

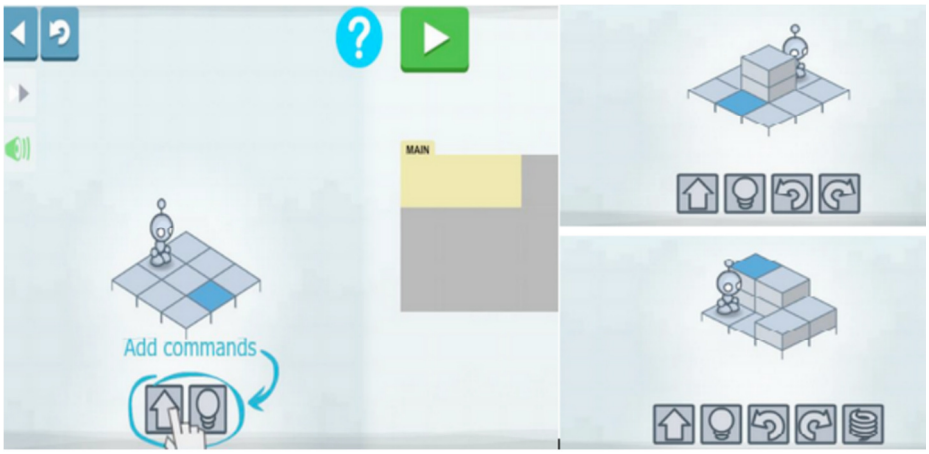


Figura 1. Interface do jogo Lightbot. Fonte: [Falcão e Barbosa 2005]

A avaliação de Lightbot foi realizada em duas etapas: a primeira é a avaliação formativa com crianças, que teve como objetivo geral analisar as possibilidades de aprendizagem por descoberta através de interação exploratória; e na segunda etapa tem-se a avaliação objetiva, a qual foi feita com base em heurísticas de usabilidade de software. Participaram onze crianças as quais contribuíram com opiniões sobre o jogo.

Como resultado, na avaliação formativa, a maioria das crianças afirmou ter gostado do jogo, entretanto, ao analisarem as heurísticas propostas, notaram vários problemas. Perceberam certa confusão das crianças ao tentar jogar, em que mesmo acessando o tutorial, as crianças não compreenderam as instruções escritas; na heurística de correspondência entre o mundo real e o sistema foi notado que as crianças tiveram uma grande dificuldade em entender que o salto leva o robô para cima e para frente, o que não foi uma ação intuitiva devido ao funcionamento da maioria de jogos de videogames. Vale destacar que diversas outras heurísticas também não foram satisfeitas.

[Santos et al. 2017] executaram o desenvolvimento e avaliação do jogo educacional Play Code Dog (Figura 2), que tem como objetivo auxiliar o ensino de lógica de Programação para crianças. Para desenvolvimento do protótipo do jogo, foram considerados alguns problemas já conhecidos em trabalhos realizados na área de jogos educacionais voltados a ensino de Programação como os jogos The Foos e Lightbot Jr. As avaliações desses trabalhos foram realizadas por [Falcão et al. 2015], mas também foram consideradas as heurísticas de jogabilidade propostas por [Barcelos et al. 2011]. Para os autores, os resultados “demonstram que o roteiro, a interface gráfica e o personagem do Play Code Dog fazem dele um jogo atraente e envolvente, capaz de motivar as crianças e cumprir bem o objetivo de ensinar de um jeito divertido”.

Figura 2. Interface do Play Code Dog. Fonte: [Santos et al. 2017]

[Silva e Cavalcanti 2018] realizaram a avaliação do Robomind (Figura 3), uma ferramenta de apoio ao ensino de Programação, com o objetivo de analisar o aprendizado de Programação dos estudantes expostos à intervenção com o Robomind e verificar se o ensino com uso da ferramenta pode diminuir a evasão em cursos de Programação. Realizaram um estudo quasi-experimental que buscou responder se o uso do Robomind no ensino de Programação para estudantes de nível médio promove mais ganhos no aprendizado em comparação a um ensino tradicional e se reduz os índices de evasão de estudantes em comparação aos que receberam aulas em um formato de ensino tradicional. Como resultado foi identificado que a ferramenta não promoveu ganhos de aprendizado significantes em relação ao grupo que não a utilizaram. Por outro lado, os estudantes do grupo experimental apresentaram menor índice de evasão, com um resultado estatisticamente significante.

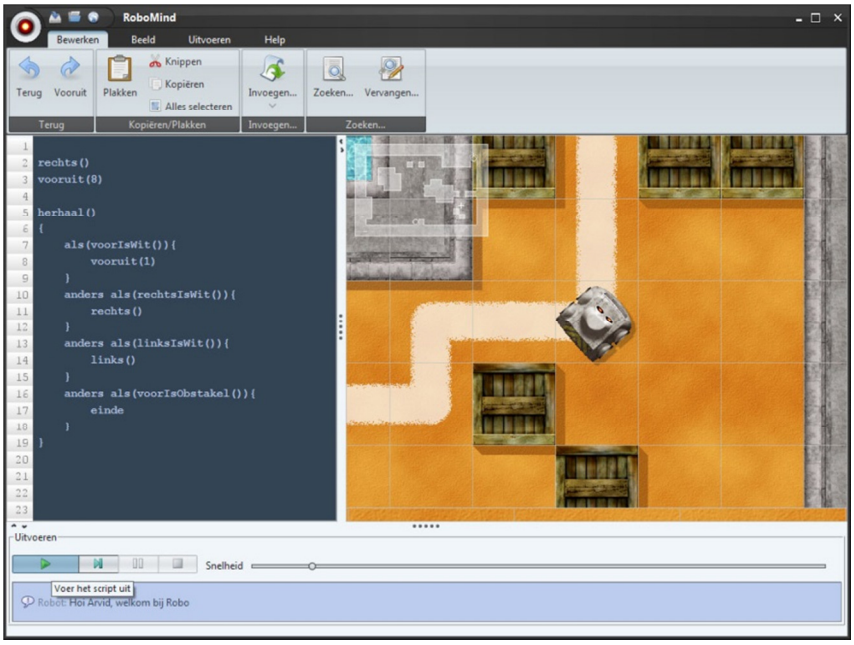


Figura 3. Interface de Robomind. Fonte: https://www.robomind.net/

[Melo et al. 2014] realizaram o desenvolvimento do jogo educacional Plant-Ação (Figura 4), que tem como objetivo auxiliar o ensino-aprendizagem das disciplinas de Introdução à Programação, de forma lúdica, fazendo com que os alunos se sintam motivados a aprender conceitos básicos de Programação, além de exercitar o raciocínio lógico enquanto progridem pelas fases do jogo. Utiliza-se o enredo em que o jogador está em um cenário onde deve cuidar de uma plantação e, para isso, deve usar blocos de ação para comandar o avatar, como, por exemplo: andar, girar, cavar, plantar, regar, cortar, colher, entre outros, bem como um específico relacionado à chamada de função. Para desenvolvimento do software os autores utilizaram Java com a biblioteca de interface gráfica javax.swing. Os autores afirmam que Plant-Ação estava em processo de aprimoramento, além de realizar um experimento em turmas do primeiro período de cursos de Bacharelado em Sistemas de Informação para validar sua eficiência.

Figura 4. Interface do Plant-Ação. Fonte: [Melo et al. 2014]

[Batista et al. 2017] desenvolveram um ambiente educacional baseado em blocos, o Poredu (Figura 5), na qual seu objetivo é ensinar comandos básicos de programação, como: sequência, condições e laços de repetição. O Poredu é destinado a crianças a partir de oito anos, e pode ser utilizada nas escolas ou em projetos de Programação e Robótica. Para o desenvolvimento do ambiente utilizaram a biblioteca Google Blockly, pela facilidade de desenvolvimento de linguagens visuais para ambiente Web. Foi realizado um experimento em que o jogo foi avaliado por crianças de 10 a 12 anos que participavam de aulas de Robótica na escola e que já haviam feito uso de ferramentas parecidas, como o Code.org. Por meio deste experimento os autores afirmaram que as crianças apontaram algumas carências na ferramenta como, por exemplo, a falta de uma barra de progresso que informasse sobre em parte da fase se encontravam; e da impossibilidade de permitir que eles resolvessem o desafio de outras maneiras, já que alcançavam o objetivo utilizando apenas uma sequência de blocos.

Além destes trabalhos, é possível encontrar outras plataformas educacionais dispostas na *Web* que são discutidos em pesquisas que abordam trabalhos como propósito de auxiliar o ensino de Programação.

[Monclar, Silva e Xéxeo 2018] apresentam Robozzle (Figura 6), um jogo de quebra-cabeças cujo objetivo é coletar estrelas dispostas em diferentes fases e níveis de dificuldades. Para isso, o jogador deve usar a lógica do uso de comandos sequenciais em blocos gráficos. Também destacam CargoBot (Figura 7), um jogo desenvolvido com propósito em ensinar a lógica de instruções em blocos de semântica facilmente identificáveis, com foco para quem não tem qualquer experiência na área de

Programação. Os desafios aumentam em níveis de dificuldade à medida que o jogador avança as fases disponíveis.

Figura 5. Interface do Poredu. Fonte: [Batista et al. 2017]

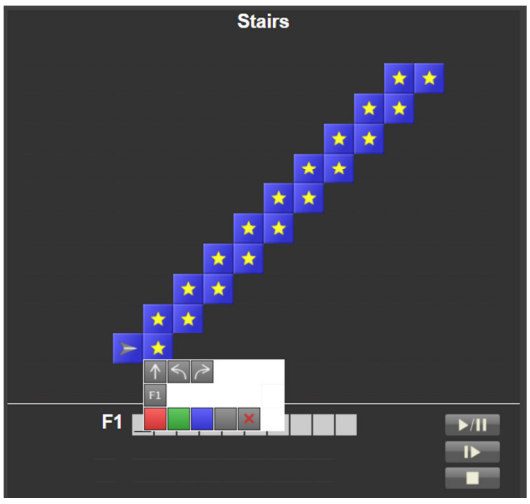


Figura 6. Interface do Robozzle. Fonte: http://robozzle.com/

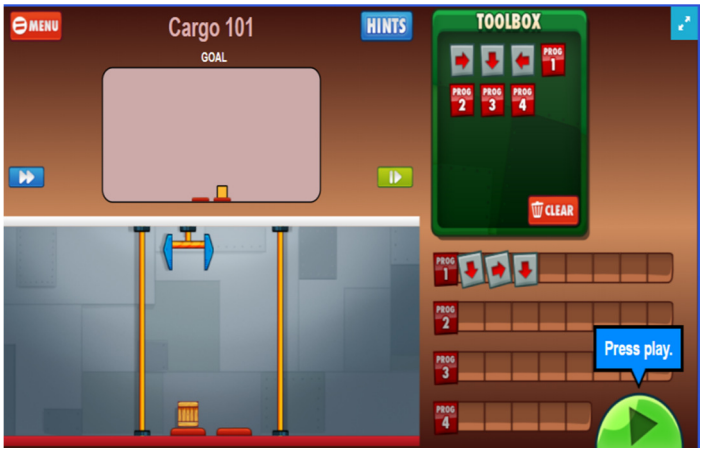


Figura 7. Interface do CargoBot. Fonte: https://altermanchess.wixsite.com/cargobot

Para melhor compreensão das características dos trabalhos relacionados e o software educacional objeto deste artigo, apresenta-se a Tabela 1.

Tabela 1. Análise Comparativa dos Trabalhos Relacionados

Característica / Software	Plataformas Disponíveis	Público-Alvo	Avaliação	Estratégia	Disponibilidade
Lightbot	Mobile (Android, Ios)	Crianças	Qualitativa e de Usabilidade	Programação em blocos	Pago
Play Code Dog	Mobile (Android)	Crianças e Iniciantes em Computação	Qualitativa	Programação em comandos	Não-disponível
Robomind	Desktop e Web	Iniciantes em Computação	Quasi-experimental	Linguagem de Programação própria	Gratuito
Plant-Ação	Desktop	Iniciantes em Computação	Não evidenciada	Programação em blocos	Não-disponível
Poredu	Web	Crianças	Qualitativa	Programação em Blocos	Gratuito (código aberto)
Robozzle	Web e Mobile	Público em Geral	Não evidenciada	Programação em Blocos	Gratuito
CargoBot	Web e Mobile	Público em Geral	Não evidenciada	Programação em Blocos	Gratuito
PaintCode	Desktop e Mobile	Público em Geral	Quali-quantitativa por Usabilidade pedagógica e design	Programação em blocos	Gratuito

Este trabalho difere dos demais por promover uma avaliação de usabilidade pedagógica e em termos de princípios de *design,* a fim de comprovar sua contribuição para o ensino de pensamento computacional além de avaliar as características contempladas em sua concepção.

Alguns problemas foram observados nos trabalhos relacionados, como a dificuldade dos utilizadores em compreender as instruções presentes no tutorial do Lightbot, o que pode demonstrar que as instruções podem não ser muito claras, por isso se faz necessário o uso de processos mais intuitivos, como a demonstração por vídeos explicativos dispostos em mídias sociais.

Em CargoBot, usa a metáfora de uma garra mecânica do tipo empilhadeira que desloca caixas no cenário, sendo assim, espera-se que essas ações fossem "compatíveis entre o sistema e o mundo real", bem como do "controle e liberdade para o usuário", sendo estes considerados conceitos de heurísticas [Nielsen 1994] recomendados em usabilidade de software. Entretanto, apesar de existir blocos para deslocar a garra para esquerda, direita e para baixo, alguns movimentos da garra devem ser abstraídos, como, por exemplo, as ações de subir a garra, agarrar uma caixa e soltá-la, tido como comandos não programáveis.

Em Robozzle, o comando de seta direcional utiliza ícone em que a direção está apontando sempre para cima (seta direcional "up"), o que nem sempre condiz com o movimento do personagem, já que este pode realizar ações de andar para direita,

esquerda ou para baixo, consequentemente, gerando dificuldade para o usuário entender a direção devida à má utilização do conceito de *affordance*, ou seja, utiliza ícone que não representa o seu significado ou ação.

Além de alguns problemas de usabilidade, observa-se que as propostas dependem de seus desenvolvedores para a criação de novos desafios, restringindo a criação de desafios e definição de exercícios por educadores com base no jogo, afetando, consequentemente, a usabilidade pedagógica dos softwares. O software desta pesquisa promove a possibilidade de criação de desafios personalizados. Apesar de Robozzle ter permitido em algum momento de seu versionamento a criação de desafios, atualmente não é possível mais realizá-la.

3. Método

3.1. Desenvolvimento

Esta pesquisa versa sobre a apresentação do software educacional PaintCode e de sua avaliação de usabilidade. Sua concepção originou-se na disciplina Interface Homem-Máquina como proposta de um ambiente em que pudesse ser utilizado para despertar o interesse em estudantes para assuntos de pensamento computacional. Por isso, a primeira premissa era de ser um ambiente que fizesse uso de um cenário lúdico e de fácil assimilação de situações presentes no cotidiano de qualquer pessoa.

Nesse sentido, o projeto de interface utilizou do enredo de uma casa que precisaria ser pintada, ou seja, versa sobre o desafio de pintar partes de um cenário incompleto em que o aluno assume o papel de um pintor e deve solucionar a problemática.

Para isso o ato de pintar precisaria utilizar uma abordagem de fácil compreensão e assimilação, como a programação baseada em blocos, em que o ato de pintar deve utilizar blocos de comandos. Portanto, a construção da solução deveria ser dada pela interação *drag-and-drop* de blocos que representam comandos de um pincel, como, por exemplo, mover para direita, mover para esquerda, mover para baixo, mover para cima, girar para a direita, girar para a esquerda, pintar, entre outras. A Figura 8 ilustra o *storyboard* da visão geral da tela de interação de PaintCode.

A codificação de PaintCode foi realizada na linguagem de programação Python [Python 2020], por permitir a criação de um software multiplataforma e da possibilidade do desenvolvimento *desktop* e *mobile* simultaneamente, através da utilização da *engine* Pygame [Pygame 2020]. Esta última auxilia no desenvolvimento e segue a licença de código aberto para colaborações. Utilizou-se a ferramenta Buildozer para compilação de versões para Android. Com isso, houve o desenvolvimento simultâneo para versão *desktop* e *mobile*.

A arquitetura de PaintCode permite extensibilidade do software com a criação de novas fases e desafios, estruturados em arquivos de texto.

A organização do código-fonte utiliza boas práticas de desenvolvimento. Definiu-se a utilização do padrão de arquitetura de software *Model, View e Controller* (MVC) [Krasner e Pope 1988], por permitir a organização do projeto, tornando-o suscetível a mudanças e aprimoramentos, como criação de *mockups* que representassem a idealização de novas fases e posteriormente a criação de suas ações e modelos.

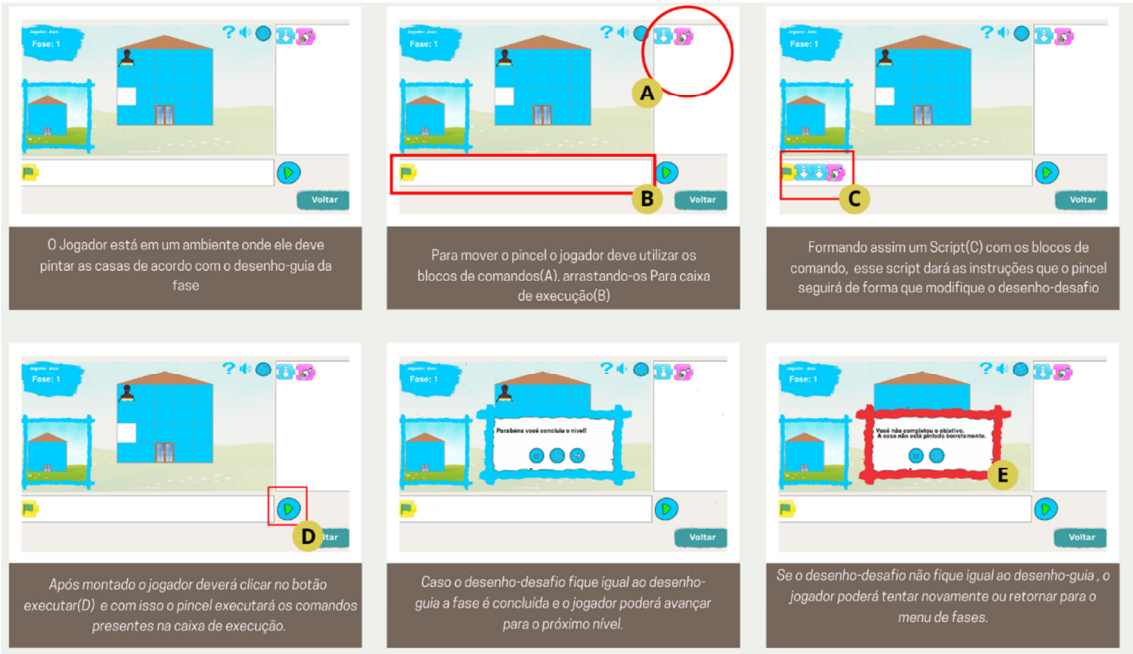


Figura 8. Projeto de interação de PaintCode dada por *storyboard*.

3.2. Avaliação de Usabilidade

Para avaliar PaintCode considera-se o fato da atual pandemia da COVID-19 [WHO 2021] que demanda ações de segurança sanitária que evitem a propagação do coronavírus em suas diversas mutações, evitando-se ações em que pessoas compartilhem o mesmo ambiente físico, devido ao isolamento social, consequentemente, não sendo possível a utilização de ambientes acadêmicos, como escolas e universidades.

Então, para viabilizar a avaliação do software, nesta pesquisa tem-se a utilização de avaliação *on-line* com método prospectivo por questionário, em formulário[2] da plataforma Google Forms, distribuído por convites enviados a e-mails de grupos de indivíduos de cursos de Computação e mídias sociais. Portanto, a escolha dos indivíduos deu-se ao acaso, uma vez que não houve controle da distribuição dos convites de participação da avaliação, implicando em uma amostragem não-probabilística e por conveniência, podendo participar dessa avaliação ergonomistas, engenheiros de software ou de usabilidade, alunos, professores, assim como outros profissionais com habilidades de avaliadores.

Para avaliação foi considerada apenas a versão *desktop* de PaintCode, pois não se teria controle dos diversos dispositivos mobiles dos usuários participantes da pesquisa, uma vez ser grande a variedade de dispositivos móveis com diferentes características, em especial a proporção de tela. Também a avaliação requer questionamentos direcionados a este tipo de tecnologia, não contempladas no planejamento desta avaliação de usabilidade.

O formulário é composto de uma breve descrição da avaliação e opção de consentimento e participação voluntária; orientações para utilização do software; tarefas a serem realizadas pelos indivíduos participantes; questionário composto de 23 questões abertas e fechadas, sendo 8 questionamentos para caracterização do perfil do usuário

[2] https://forms.gle/EHTpFahyKFbfMszC7

(Tabela 2), 10 questões para usabilidade de *design* (Tabela 3) e 5 questões sobre usabilidade pedagógica (Tabela 4) que consideram aspectos motivacionais; além de campos para comentários e sugestões e atribuição de uma nota de experiência geral a partir da experiência da utilização do software.

As questões de usabilidade (Tabelas 3 e 4) estão escritas de maneira positivo-afirmativa, uma vez que os pesquisadores desta pesquisa entendem que dessa forma a assimilação entre concordância e discordância é mais natural. As respostas dessas questões estão dispostas em escala Likert [Likert 1932] como métrica da avaliação, com opções para Discordo totalmente (DT), Discordo (D), Neutro (N), Concordo (C) e Concordo totalmente (CT).

Para a elaboração das questões, tomou-se por base trabalhos presentes na literatura [Azevêdo et al. 2018, Scherer 2018, Gouvêa e Nakamoto 2015, Medeiros 2014, Shoukry et al. 2014, Krone 2013, Valle et al. 2013, Lopes 2012, Ergolist 2011], pois se tem o entendimento que as questões já foram previamente validadas, eliminando a necessidade de realização de um pré-teste. Todas as questões refletem o planejamento do projeto de interface do software, portanto não se abordam outros princípios não contemplados. As questões de usabilidade de *design* refletem princípios baseados a partir de heurísticas de Nielsen [Nielsen 1993, Nielsen 1994].

Na avaliação um usuário inspeciona a usabilidade, sendo este um tipo de avaliação de baixo custo e alto benefício em a interface é examinada verificando as propriedades de usabilidade definidas [Ferreira et al. 2014]. Para inspeção do percurso cognitivo, em que é avaliada a aprendizagem por exploração, antes de responder as questões relacionadas à avaliação do software, no formulário foi previamente proposta a realização de algumas tarefas (T), a saber: T1 – realize o download de PaintCode; T2 – descompacte o arquivo. Execute o arquivo PaintCode.exe; T3 – configure a velocidade das animações de PaintCode; T4 – clique em começar; T5 – conclua os níveis 1, 2, 3, 4, 5; T6 – volte para a tela inicial; T7 – crie um novo progresso; T8 – exclua o progresso criado na tarefa anterior; T9 – continue com os níveis 6, 7, 8; T10 – volte para a tela inicial, clique na opção CRIAR e crie uma Fase Personalizada; e T11 – acesse a opção CRIAÇÕES do menu COMEÇAR e entre na Fase criada por você anteriormente.

Tabela 2. Questionário de caracterização de perfil de usuário

Nº	Questões
Q1	Qual sua faixa etária?
Q2	Qual sua escolaridade?
Q3	Qual seu sexo?
Q4	Qual a sua ocupação?
Q5	Qual sua área de atuação?
Q6	Qual seu nível de conhecimento sobre conceitos de Algoritmo?
Q7	Quais dispositivos você costuma utilizar softwares que auxiliam na prática de estudos de Programação?
Q8	Com que frequência você costuma utilizar softwares para auxiliar na prática de estudos de Programação?

Tabela 3. Questionário de usabilidade de *design*

Nº	Questões	Heurísticas	Autores
Q9	O PaintCode é de fácil utilização.	Facilidade de aprendizagem	[Scherer 2018]
Q10	NÃO é preciso aprender uma série de comandos para utilizar o PaintCode.		
Q11	Os elementos de PaintCode são capazes de evitar que o usuário execute uma ação acidentalmente.	Minimização de erros	[Azevêdo et al. 2018]
Q12	Há informações de ajuda suficientes para que o usuário utilize o PaintCode.		
Q13	Os ícones são distintos uns dos outros e possuem sempre o mesmo significado de uma tela para outra.	Consistência	[Ergolist 2011]
Q14	A organização em termos da localização das várias características das janelas é mantida consistente de uma tela para outra.		
Q15	São exibidas apenas opções relacionadas as ações que se podem realizar em uma tela.	Estética e design minimalista	[Krone 2013]
Q16	O PaintCode exibe quantidades pequenas de informações em cada tela, sem texto ou imagens em excesso.		
Q17	Os elementos de PaintCode permitem que o usuário realize suas ações de forma eficiente, ou seja, com menor esforço possível.	Eficiência	[Valle et al. 2013]
Q18	A interação com o PaintCode é ágil, podendo-se acessar rapidamente as telas e menus.		[Shoukry 2014]

Tabela 4. Questionário de princípios pedagógicos

Nº	Questões	Heurísticas	Autores
Q19	O PaintCode desperta no usuário interesse pelo conteúdo.	Aspectos Motivacionais	[Gouvêa e Nakamoto 2015]
Q20	O PaintCode apresenta desafios pedagógicos adequados ao aprendizado do tema abordado.		
Q21	O feedback que o aluno recebe é rápido e estimula o raciocínio para encontrar a resposta correta.		
Q22	O uso de PaintCode contribui para o aprendizado.		[Lopes 2012]
Q23	O PaintCode é atrativo, envolve e motiva para o desafio de concluir o jogo.		[Medeiros 2014]

Antes de iniciar a avaliação de PaintCode, inicialmente verificou-se a qualidade das respostas através do coeficiente (αC) alfa de Cronbach [Cronbach 1951,George e Mallery 2003, p. 231]. [Cortina 1993] considera uma das ferramentas estatísticas mais importantes e difundidas em pesquisas envolvendo a construção de testes. É tido como um dos indicadores mais importantes da qualidade de uma escala de medida, no qual pode variar entre 0 e 1. [George e Mallery 2003, p. 231] apontam a seguinte relação de qualidade de dados para os valores de αC: $\alpha C > 0.9$ é excelente; $\alpha C > 0.8$ é bom; $\alpha C > 0.7$ é aceitável; $\alpha C > 0.6$ é questionável; $\alpha C > 0.5$ é ruim; e $\alpha C < 0.5$ é inaceitável. Entretanto, para [Landis e Koch 1977] a qualidade de consistência interna para αC é: $\alpha C < 0,21$ é pequena; $0,21 < \alpha C < 0,40$ é razoável; $0,41 < \alpha C < 0,60$ é moderada; $0,61 < \alpha C < 0,80$ é substancial; e $\alpha C > 0,80$ é quase perfeita.

Em seguida, tem-se: (i) análise descritiva dos dados; (ii) para avaliação de qualidade de software: System Usability Scale (SUS) [Bangor, Kortum e Miller 2009]; e (iii) inferência estatística a partir de testes de hipóteses por Análise de Variância ANOVA [Vieira 2006] e Qui-Quadrado [Plackett 1983 apud Levin e Fox 2004].

O SUS utiliza um questionário com 10 questões que refletem princípios de *design*, alternadas entre afirmativas e negativas, com respostas dispostas em escala [Likert 1932], para averiguar o nível de usabilidade de um sistema [Brooke 1996].

Uma das características do SUS é a de que o questionário utiliza perguntas ímpares escritas na forma afirmativa e pares na negativa. Porém, como as questões deste trabalho foram dispostas de maneira afirmativo-positiva, então se inverteu o valor das respostas das perguntas pares, ou seja, as respostas com valor 1 (discordo totalmente) se transformaria em 5 (concordo totalmente), 2 (discordo) em 4 (concordo), e assim sucessivamente. Com isso, reflete-se a opinião inversa caso uma questão estivesse escrita de maneira positivo-negativa.

Visando a imparcialidade na análise e comparações para todas as análises que se seguem, são consideradas as respostas dos indivíduos as mesmas utilizadas em SUS.

Por conseguinte, aplicou-se o algoritmo SUS [Brooke 1996], ilustrado na Figura 9, para obtenção da pontuação do software, a saber: (i) nas questões ímpares a pontuação da resposta da pergunta corresponde a posição na escala menos 1; (ii) nas pares a pontuação corresponde a posição na escala subtraída de 5; (iii) em seguida, para cada indivíduo participante somam-se as pontuações de cada questão e depois multiplica-a por 2,5; e, por fim, (iv) obtêm-se a média das pontuações de todos os indivíduos, sendo esta a pontuação final.

[Bangor, Kortum e Miller 2009] estabelecem alguns critérios para a pontuação SUS (Figura 10): faixas de quartil, faixas de acessibilidade e adjetivos de classificação. No primeiro critério são quatro quartis, em que o primeiro define a usabilidade do software faixa de acessibilidade como baixa, ou seja, inaceitável; a partir do segundo quartil o software está com faixa de acessibilidade marginal alta, em direção da aceitação. A acessibilidade inaceitável é quando a pontuação SUS do software está abaixo de 50, apontando a necessidade de melhorias. As subescalas de adjetivações de usabilidade apontam para pior imaginável quando a pontuação é de até 25 pontos, pobre para pontuações entre 25 a 39 pontos, ok para acima de 39 a 52 pontos, mas ainda com acessibilidade inaceitável o que aponta indícios de potenciais problemas, bom entre 53 e 73 pontos, excelente para acima de 73 e abaixo de 85 pontos, a partir de 85 pontos a usabilidade é tida como melhor imaginável. Destaca-se a possibilidade com adjetivação

boa poder estar em uma faixa de acessibilidade marginal, portanto sugere-se que a pontuação seja de pelo menos 70 pontos para que além de bom seja classificado com acessibilidade aceitável.

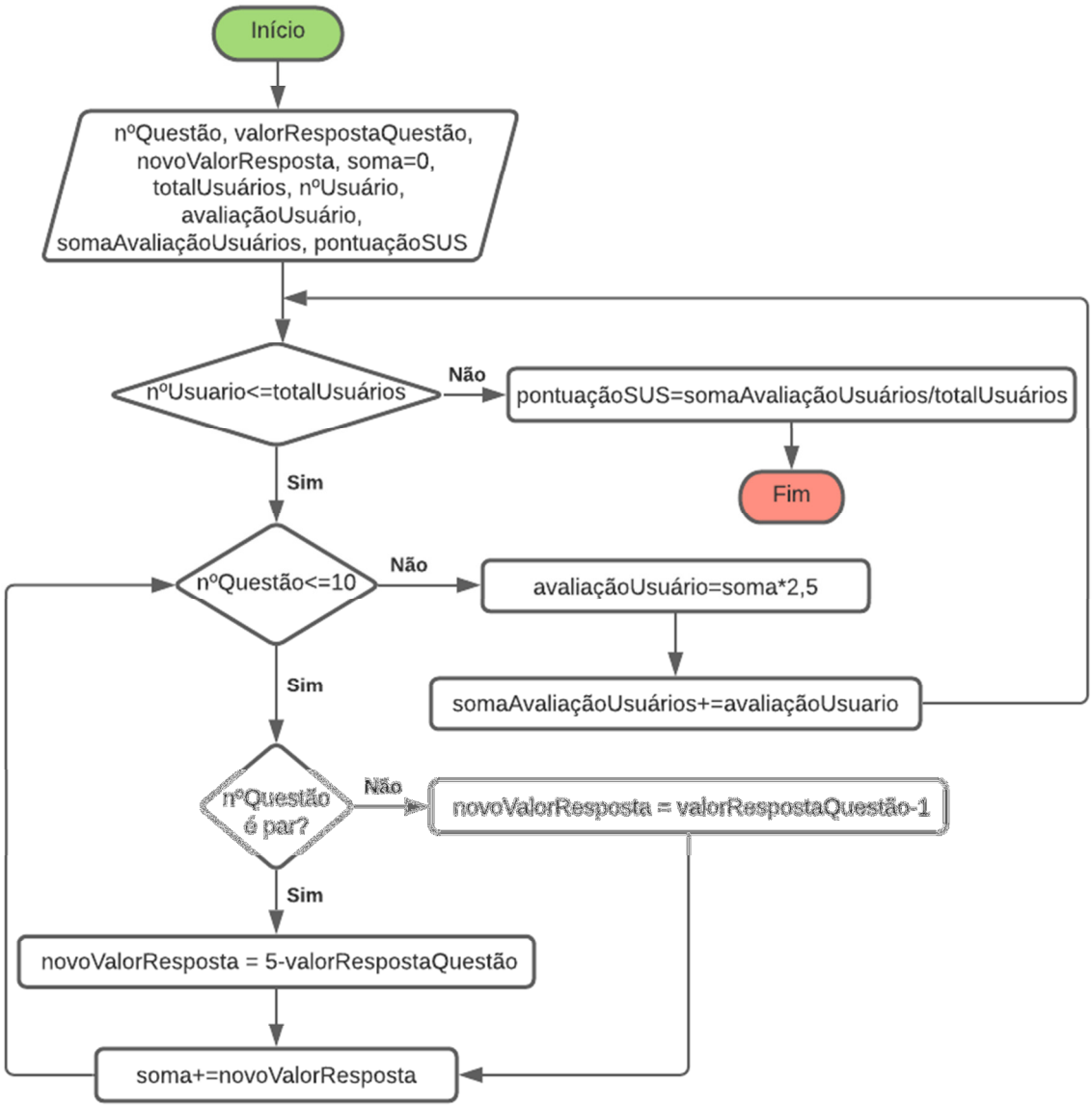


Figura 9. Fluxograma para obtenção de pontuação SUS desta pesquisa

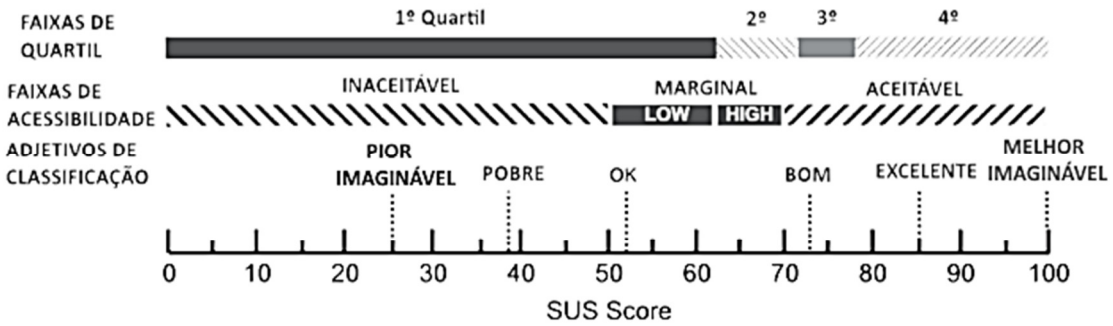


Figura 10. Escalas de avaliação de usabilidade por SUS
Fonte: Traduzido de [Bangor, Kortum e Miller 2009]

Para a análise por inferência estatística na avaliação de PaintCode tem-se a verificações de pressupostos por testes de hipóteses, chamadas de hipótese nula (H_0) e alternativa (H_1), que retratam a avaliação do software pretendida. Para essa avaliação são averiguadas três situações: (i) Situação 1 (S1) – do ponto de vista de qualidade do software; (ii) Situação 2 (S2) – da relação de dependência de variáveis de caracterização de indivíduo em relação à avaliação do software; e (iii) Situação 3 (S3) – em relação à contribuição do software, do ponto de vista de sua usabilidade, como ferramenta para auxiliar no processo de ensino e aprendizagem de conceitos básicos de Programação.

A ANOVA a partir do Teste F será utilizada para responder a Situação 1, a partir da comparação da distribuição de vários grupos em amostras independentes. O teste paramétrico compara simultaneamente duas ou mais médias, ou seja, permite testar hipóteses sobre médias de distintas populações e é sobre estas subpopulações que é construída a hipótese a ser testada [Vieira 2006].

Para a primeira situação, será verificada se há diferença significativa entre as médias (μ) das avaliações de percepção de Experiência Geral (EG) em relação a SUS. Experiência Geral é tida como uma avaliação livre em que o indivíduo atribui uma nota a partir de sua experiência com a utilização de PaintCode. Sendo assim, têm-se as seguintes hipóteses: (i) $H_{0\text{-}S1}$: $(\mu_{EG} - \mu_{SUS}) = 0$. As médias das notas atribuídas pelos indivíduos para Experiência Geral e calculadas por SUS são consideradas estatisticamente iguais; e (ii) $H_{1\text{-}S1}$: $(\mu_{EG} - \mu_{SUS}) \neq 0$. As médias das notas atribuídas pelos indivíduos para Experiência Geral e calculadas por SUS diferem estatisticamente.

Para analisar se há relação de dependência de variáveis de caracterização de indivíduo em relação à avaliação de PaintCode, pretende-se verificar na segunda situação as seguintes hipóteses: (i) $H_{0\text{-}S2}$: avaliação do software independe de variável de caracterização de indivíduo; e (ii) $H_{1\text{-}S2}$: avaliação do software é dependente de variável de caracterização de indivíduo.

Define-se que se há relação de dependência entre a avaliação de usabilidade do software e o grau de aceitação dos aspectos definidos desta usabilidade e se observada positividade em relação a esses aspectos, então a usabilidade do software contribui para o objetivo ao qual foi desenvolvido. Portanto, verifica-se a terceira situação, através das hipóteses: (i) $H_{0\text{-}S3}$: o grau de usabilidade de PaintCode não fornece contribuição significante para o processo de ensino e aprendizagem de conceitos básicos de Programação; e (ii) $H_{1\text{-}S3}$: o grau de usabilidade de PaintCode fornece contribuição significante para o processo de ensino e aprendizagem de conceitos básicos de Programação.

Para as Situações 2 e 3 será utilizado o teste hipóteses por Qui-Quadrado, mas as tabelas de contingência tabelas *r x k*, (várias linhas x várias colunas), o teste Qui-quadrado pode ser aplicado somente se o número de células com frequência esperadas inferior a 5 é 20% do total das células e se nenhuma célula tem frequência esperada inferior a 1. Se essas condições não são satisfeitas pelos dados na forma em que foram coletados originalmente, o pesquisador deve combinar categorias adjacentes de modo a aumentar as frequências esperadas nas diversas células [Siegel 1975, p. 124].

O detalhamento de todos os métodos e resultados obtidos da pesquisa é encontrado em [Silva e D'Emery 2021]

4. Resultados e Discussões

4.1. O Software Educacional PaintCode

PaintCode[3] é apresentado na Figura 11, na qual ilustram-se as principais telas.

Na Figura 11a tem-se a tela abertura de PaintCode, onde será possível começar um novo progresso ou acessar perfis criados (Figura 11c). Selecionando um perfil, o usuário seguirá para tela de seleção de fase (Figura 11d). As Figuras 11e e 11f são exemplos de fases, nas quais se pode observar o ambiente do software educacional baseado em programação em blocos e comandos para sequências (avançar para cima, para baixo, para esquerda e direta, girar 90º para esquerda e para direita); variável (pincel); estrutura de seleção (opção de cor), laço de repetição e chamada de função (*fn*), argumentos, linhas de comandos e execução. Na Figura 11b, ilustra a configuração de velocidade de animação em que se executa uma sequência de comandos definida. Essas ações são apresentadas em animação, retratando a execução da interpretação de cada comando. A criação de fases e fases criadas é ilustrada nas Figuras 11g e 11h, respetivamente.

O armazenamento de cada fase é dado através de um arquivo (Figura 12a) estruturado, contendo informações para o desenho das respostas (Figura 12b) e desenho desafio (Figura 12c). Cada cor e objeto disponíveis são representados por um número, por exemplo: a janela, tida como um obstáculo, é representada por -1, a cor branca por 0, azul por 1, e assim sucessivamente. Na Figura 12a é possível observar que a linha 2 corresponde a Figura 12b, enquanto a linha 3 a Figura 12c. A linha 1 do arquivo (Figura 12a) corresponde as cores do desafio, por exemplo: 0-branco, 1-azul, 2-verde e 3-amarelo. Esse arquivo é encontrado na pasta pessoal do usuário, podendo ser utilizado inclusive em futuras versões do software. Dados sobre as fases personalizadas e progredidas (Figura 13b) também são salvos em arquivo (Figura 13a).

4.2. Análise de Dados e Avaliação de PaintCode

Participaram da avaliação de PaintCode um total de 36 indivíduos, dos quais 72,2% (26/36) são do sexo masculino e 27,8% (10/36) feminino; possuem idade entre 20 e 29 anos (75% - 27/36), 30 anos ou mais (16,7% - 6/36), 16 a 19 anos (8,3% - 3/36); são estudantes de graduação (61,1% 22/36), mas observa-se a participação de 8 indivíduos com ensino superior completo, 3 especialistas/mestres/doutores, 2 com ensino médio completo e 1 com ensino médio incompleto; a maioria dos indivíduos são alunos (67% - 24/36), 14% (5/36) são professores e 19% (7/36) são de outras atividades; 75% (27/36) dos participantes afirmaram ser da área de Computação e apenas 25% (9/36) são de outras áreas; e sobre a frequência de uso de softwares para estudo de Programação: apenas 11% (4/36) afirmam sempre utilizar, 22% (8/36) frequentemente, 25% (9/36) às vezes, 28% (10/36) raramente e 14% (5/36) nunca.

[3] https://linktr.ee/paintcode

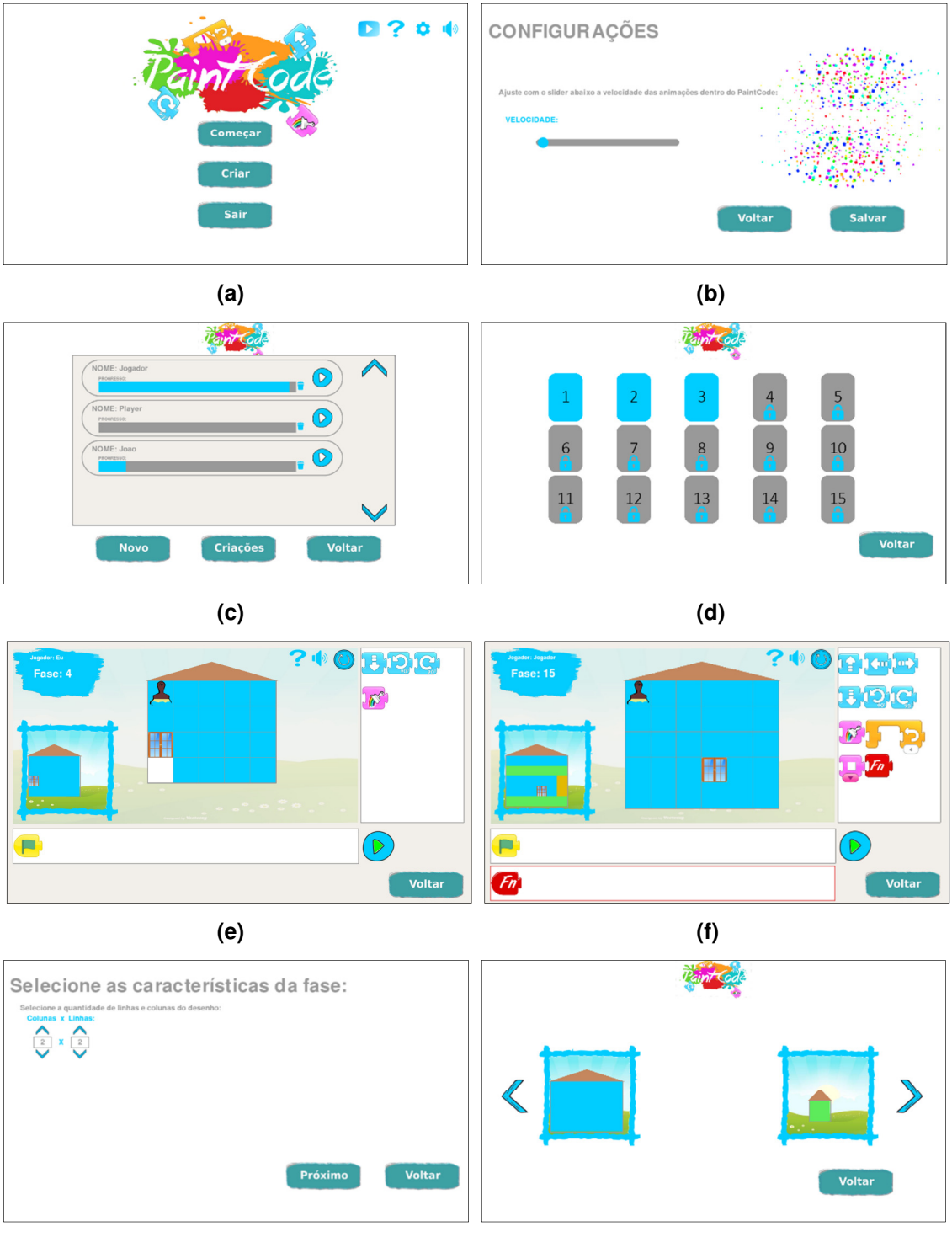
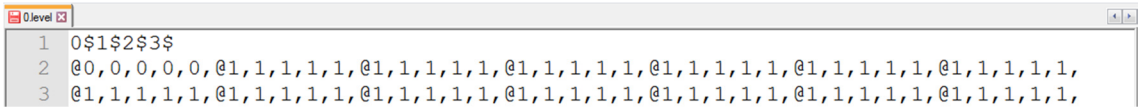
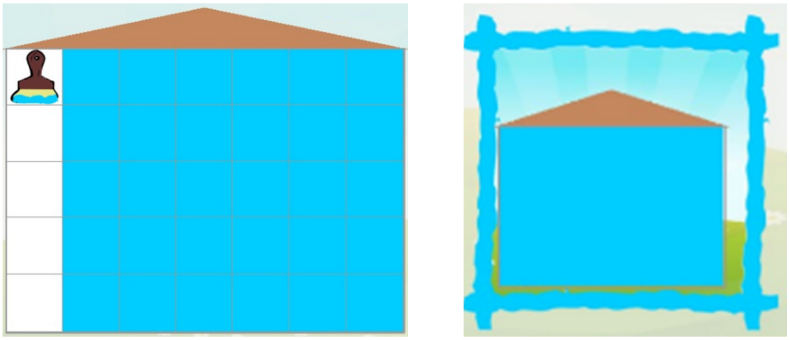


Figura 11. Interfaces de PaintCode: (a) inicial; (b) configuração de velocidade; (c) perfil; (d) fases; (e) e (f) exemplo de fase; (g) criação de fase; e (g) fases personalizadas

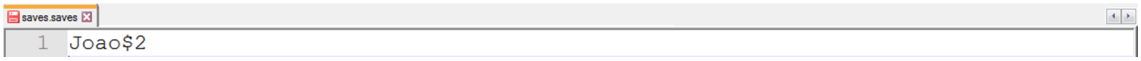


(a)

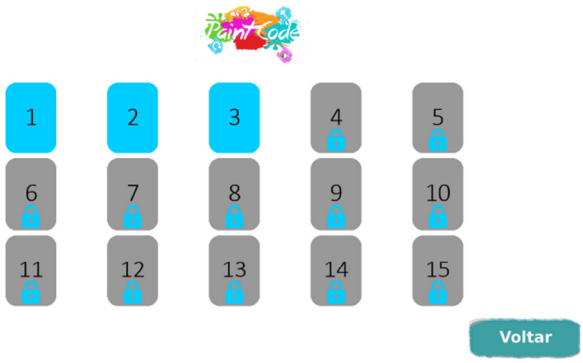


(b) (c)

Figura 12. Armazenamento de fases: (a) arquivo estruturado; (b) fase desafio; (c) fase objetivo



(a)



(b)

Figura 13. Armazenamento de perfil: (a) arquivo estruturado; e (b) seleção de fases

Antes de iniciar a avaliação de PaintCode, inicialmente verifica-se a qualidade das respostas. A Tabela 5 apresenta os αC obtidos.

Tabela 5. Alfa de Cronbach para os dados da avaliação

Usabilidade	Nº de Questões (*n*)	Nº de Indivíduos	Total de respostas	α	[George e Mallery 2003]	[Landis e Koch 1977]
Princípios de design	10	36	360	0,64	Questionável	Substancial
Pedagógica	5	36	180	0,74	Aceitável	Substancial
Todas as Questões	15	36	540	0,70	Aceitável	Substancial

Portanto, afirma-se que os dados levantados por meio do questionário (Tabelas 3 e 4) são tidos como um instrumento de medida confiável.

A pontuação SUS obtida por PaintCode é de 91,38 pontos, o que significa uma adjetivação para o software de "Melhor imaginável". Vale ressaltar, que a média de percepção de Experiência Geral atribuída pelos indivíduos participantes da avaliação foi de 95 pontos. A Tabela 6 sintetiza as pontuações.

Tabela 6. Comparativo das pontuações de Experiência Geral e SUS

Avaliação	Pontuação
SUS	91,38
Experiência Geral	95,00

As médias das avaliações de Experiência Geral e SUS são muito próximas, levando a análise das hipóteses definidas na Situação 1 ($H_{0\text{-}S1}$ e $H_{1\text{-}S1}$), ou seja, verificar se as médias possuem diferença estatisticamente significante. A Tabela 7 apresenta os resultados da ANOVA.

Tabela 7. ANOVA para os valores médios de Experiência Geral e SUS

Anova: fator único						
Grupo	**Contagem**	**Soma**	**Média**	**Variância**	-	
SUS	36	3290	91,389	80,516		
Experiência Geral	36	3420	95	60		
Fonte da variação	**SQ**	**gl**	**MQ**	**F**	**valor-P**	**F crítico**
Entre grupos	234,722	1	234,722	3,3409	0,072	3,9778
Dentro dos grupos	4918,056	70	70,258	-	-	-
Total	5152,778	71	-	-	-	-
Legenda: SQ(s^2) - soma dos quadrados, gl - grau de liberdade, MQ(s^2) - quadrado médio						

Na comparação entre Experiência Geral e SUS assume-se o nível de significância (α) de 5%, logo como na ANOVA o valor-P < α ($p<0,05$), logo $H_{0\text{-}S1}$ não é refutada, ou seja, as médias das notas atribuídas pelos indivíduos para Experiência Geral e calculadas por SUS são consideradas estatisticamente iguais, pois as médias atribuídas nas avaliações não diferem estatisticamente. Logo as avaliações apontam similaridade de percepção.

As implicações de dependência definidas na Situação 2, ou seja, da relação de dependência de variáveis de caracterização de indivíduo em relação à avaliação de usabilidade de PaintCode. São as variáveis: **sexo, faixa etária, área de atuação, escolaridade, nível de conhecimento em algoritmos, uso de softwares educacionais para auxiliar nos estudos de Programação**. Essas variáveis são correspondentes às questões de caracterização de indivíduo.

Para definir as dependências estatísticas entre as variáveis de caracterização de indivíduo e a usabilidade, foram realizadas as seguintes combinações a partir das variáveis: (i) idade possui 2 categorias ($16 \leq$ idade ≥ 29 e idade ≥ 30) e 5 níveis de concordância; (ii) nível de conhecimento possui duas categorias (conhece com níveis 3 a 5 e pouco com níveis 1 e 2) e 3 níveis de concordância; (iii) escolaridade com 3 categorias (ensino médio, ensino superior incompleto, ensino superior completo com pós-graduação) e 2 níveis de concordância; e (iv) utilização de softwares educacionais para auxiliar nos estudos de Programação com três categorias (às vezes com raramente, frequentemente com sempre, nunca) e 2 níveis de concordância.

Os 3 níveis de concordância definidos na combinação de categorias, têm-se: discordância (combinação de discordo totalmente e discordo), neutro e concordância (combinação de concordo totalmente e concordo); quando 2 níveis: discordância (combinação de discordo totalmente e discordo) e concordância (combinação de concordo totalmente e concordo e neutro).

Na Tabela 8 é apresentado um resumo das análises de dependência das variáveis. A marcação nas células com "Dependente" configura que foi identificado uma dependência entre a variável e a usabilidade, ou seja, no teste Qui-Quadrado o valor calculado de p é inferior a 0,05 (p<0,05). Portanto, conclui-se que a avalição de usabilidade independe da caracterização do indivíduo; pois em todas as variáveis a hipótese nula ($H_{0\text{-}S2}$) não são refutadas.

Tabela 8. Efeito das variáveis de caracterização de indivíduo sobre as de usabilidade

Variável	Usabilidade	
	***p* (teste χ^2)**	**Análise**
Idade	0,475	Não dependente
Sexo	0,075	Não dependente
Área de atuação	0,983	Não dependente
Escolaridade	0,883	Não dependente
Nível de conhecimento em algoritmos	0,982	Não dependente
Uso de softwares educacionais para auxiliar nos estudos de Programação	0,940	Não dependente

Para análise da Situação 3, a Tabela 9 apresenta as frequências observadas, enquanto na Tabela 10 decorre-se o cálculo da estatística χ^2.

Tabela 9. Frequências observadas

	Usabilidade		
Métricas	**Pedagógica**	***Design***	**Total**
Discordo totalmente	0	141	**141**
Discordo	0	30	**30**
Neutro	2	12	**14**
Concordo	27	38	**65**
Concordo totalmente	151	139	**290**
Total	**180**	**360**	**540**

Para a análise de inferência estatística da Situação 3, obteve-se $\chi^2_{calculado}$ = 135,564. Como $\chi^2_{tabelado}$ = 9,488 (gl = 4 ; α = 5%), então se refuta $H_{0\text{-}S3}$, pois $\chi^2_{calculado}$ $\chi^2_{tabelado}$ > $\chi^2_{tabelado}$. Logo, a hipótese nula ($H_{0\text{-}S3}$) é refutada, ou seja, o grau de usabilidade de PaintCode fornece contribuição significante para o processo de ensino e aprendizagem de conceitos básicos de programação ($H_{1\text{-}S3}$), tendo em vista a forte dependência (p<0,01) entre as variáveis de usabilidade e o grau de aceitação, além do fato do nível de aceitação ser alto para as afirmações definidas no questionário de avaliação, conforme apresentado nas Figuras 14, 15 e 16, as quais mostram a distribuição dos níveis de concordância, positivo afirmativo, às respostas relativas às usabilidades de *design* e pedagógica, à distribuição das respostas para cada heurística avaliada e às respostas as questões de usabilidade pedagógica, respectivamente.

Tabela 10. Cálculo de χ2 para usabilidade

Métricas	f_o	f_e	$(f_o - f_e)$	$(f_o - f_e)^2$	$\frac{(f_o - f_e)^2}{f_e}$	χ^2_{UP}	χ^2_{UD}
DT (UP)	0	47,00	-47,00	2209,00	47	**90,376**	**45,188**
DT (UD)	141	94,00	47,00	2209,00	23,5		
DT (UP)	0	10,00	-10,00	100,00	10		
DT (UD)	30	20,00	10,00	100,00	5		
N (UP)	2	4,67	-2,67	7,11	1,524		
N (UD)	12	9,33	2,67	7,11	0,76		
C (UP)	27	21,67	5,33	28,44	1,313		
C (UD)	38	43,33	-5,33	28,44	0,66		
CT (UP)	151	96,67	54,33	2952,11	30,539		
CT (UD)	139	193,33	-54,33	2952,11	15,26		
					χ^2	**135,564**	

Legenda: DT – Discordo Totalmente; D – Discordo; N – Neutro; CT – Concordo Totalmente; C – Concordo; UP – usabilidade Pedagógica; UD – Usabilidade de Design.

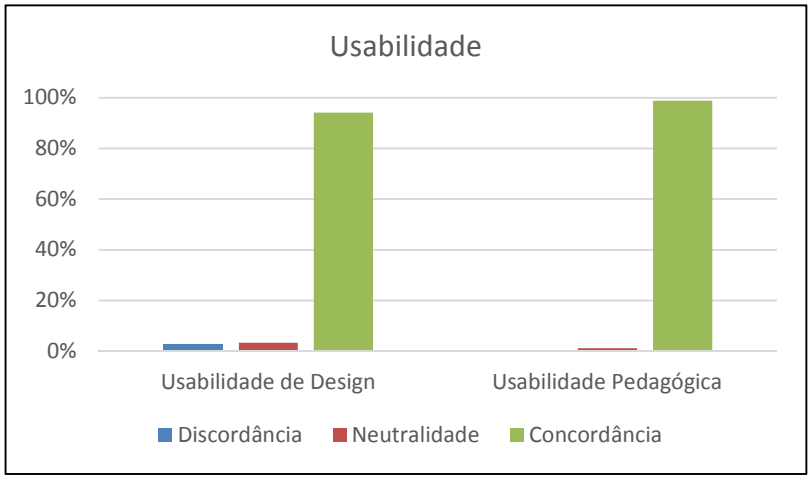


Figura 14. Níveis de concordância por usabilidade

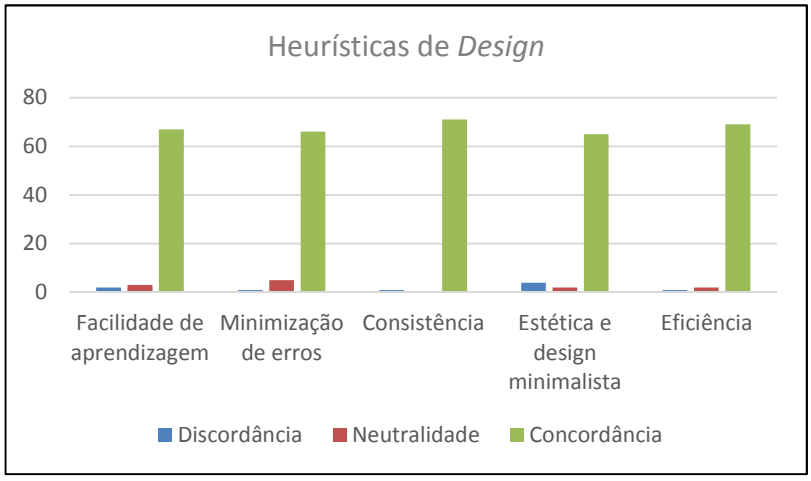


Figura 15. Níveis de concordância das heurísticas de usabilidade de design

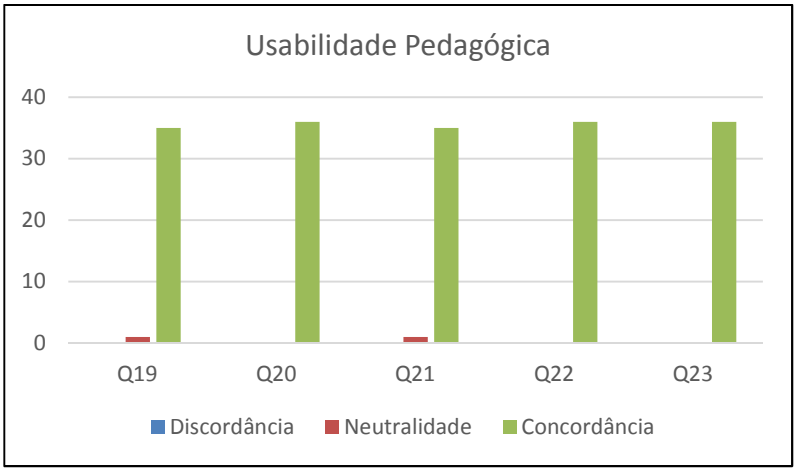


Figura 16. Níveis de concordância de questões de usabilidade pedagógica

5. Conclusão

Ao fazer uma análise sobre o índice de reprovação e desistência de cursos de Computação, percebe-se o quão são necessárias pesquisas para promover soluções que possam ser utilizadas como ferramentas auxiliares no processo de ensino e aprendizagem de assuntos de turmas iniciais.

Uma das principais dificuldades encontradas nesta pesquisa foi a da pandemia da COVID-19, que impossibilitou a realização de um experimento controlado. Por este motivo foi realizada uma avaliação de forma on-line, que dificultou a observação direta dos participantes através de um ambiente controlado, a fim de obter mais *feedbacks* e analisar as reações e dificuldades dos participantes.

A avaliação apontou excelente desempenho de PaintCode, tanto na avaliação de qualidade de SUS com pontuação 91,38, quanto pelas hipóteses analisadas por inferência estatística. Entretanto, sabe-se que a utilização de amostragem feita por conveniência, ou seja, através de uma amostra de pequeno grupo de indivíduos que estão dentro de um contexto, pode ter como consequência a incapacidade de generalizar os resultados para toda população, mas esse tipo abordagem foi necessário, uma vez que a pesquisa se decorre em um período atípico que inviabiliza o contato direto com usuários. Ainda assim, acredita-se que a abordagem utilizada é válida e não desqualifica a pesquisa realizada.

A partir dos comentários/sugestões, diversos *feedbacks* positivos foram dados sobre PaintCode. Mas algumas sugestões poderão ser consideradas para melhoria do software em trabalhos futuros, como, por exemplo: possibilitar a criação de novas fases a partir de outros objetos (carros, frutas e objetos diversos); criar mais fases; possibilitar uma transição mais suave entre as dificuldades das fases; comando para desfazer uma ação; por fim, na criação de fases maiores deve-se permitir a inclusão de mais blocos de comandos para resolver o desafio. Além disso, como trabalho futuro pretende-se realizar a avaliação da versão *mobile* de PaintCode, e também uma comparação entre as duas versões (*desktop* e *mobile*) afim de validar a interação com a versão mobile.

Referências

Alves, P. e Pires, J. A. (2002). "A usabilidade em software educativo: princípios e técnicas", https://bibliotecadigital.ipb.pt/bitstream/10198/1950/1/2002_A%20usabilidade%20em%20Software%20educativo%20IE.pdf, Setembro.

Azevêdo, M., Rousy, D., Siebra, C. (2018). AHJED - Avaliação Heurística para Jogos Educacionais Digitais. In: Nuevas Ideas En Informática Educativ, v.14, n.1, p.126-136.

Bangor, A., Kortum, P. e Miller, J. (2009). Determining what individual SUS scores mean: adding an adjective rating scale. In: Journal of Usability Studies, 4 ed, p.114-123.

Barcelos, T. S., Carvalho, T., Schimiguel, J. e Silveira, I. F. (2011). Análise comparativa de heurísticas para avaliação de jogos digitais. In: Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction. p. 187-196.

Batista, E. J. S., Silva, L., Leite, C., e Lima, A. (2017). Poredu: um ambiente de programação em blocos. In: Anais Dos Workshops Do VI Congresso Brasileiro de Informática Na Educação (CBIE 2017), p. 144.

Bezerra, F. e Dias, K. P. (2014). Programação de Computadores no Ensino Fundamental: Experiências com Logo e Scratch em escola pública. In: Anais do XXII Workshop sobre Educação em Informática, Brasília p. 229-238.

Brasil. (2020). Sinopse de estatística de ensino superior, Instituto Nacional de Estudos e Pesquisa Educacionais Anísio Texeira (INEP), https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-da-educacao-superior/resultados

Brasscom. (2020). Mercado de TI tem grande demanda e déficit de novos profissionais. Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação (Brasscom), https://brasscom.org.br/mercado-de-ti-tem-grande-demanda-e-deficit-de-novos-profissionais/, Março.

Brooke, J. (1996). SUS: A "quick and dirty" usability scale. In: Jordan, P. W., Thomas, B., Weerdmeester, B. A. and McClelland, A. L. (Eds.), Usability Evaluation in Industry. London: Taylor and Francis.

Cavalcante, A., Costa, L. e Araujo, A. L. (2016). Um Estudo de Caso Sobre Competências do Pensamento Computacional Desenvolvidas na Programação em Blocos no Code.Org. In: Anais Do V Workshop sobre Congresso Brasileiro de Informática Na Educação (CBIE 2016), p. 1117.

Cortina, J. M. (1993). What is coefficient alpha? An examination of theory and applications. In: Journal of Applied Psychology. v. 78, p. 98-104.

Cronbach, J. L. (1951). Coefficient alpha and the internal structure of tests. In: Psychometrika, v. 16. n. 3, p. 297-334.

Falcão, T. e Barbosa, R. (2015). "Aperta o Play!" Análise da Interação Exploratória em um Jogo Baseado em Pensamento Computacional. In: Anais do XXVI SBIE*, v.*26, n.1, p.419-428.

Ergolist. (2011). "ErgoList - LabIUtil - UFSC", http://www.labiutil.inf.ufsc.br/ergolist/index.html/, Setembro.

Ferreira, B. M., Rivero, L., Lopes, A., Marques, A. B. e Conte, T. (2014). "UsabiliCity: Um jogo de apoio ao ensino de propriedades de usabilidade de software através de analogias". In: Anais do Simpósio Brasileiro de Informática na Educação.

George, D. and Mallery, P. (2003). SPSS for Windows step by step: A simple guide and reference. 11.0 update (4th ed.). Boston: Allyn & Bacon. In: Gliem, J. A., Gliem, R. R. (2003). Calculating, Interpreting, and Reporting Cronbach's Alpha Reliability Coefficient for Likert-Type Scales, Midwest Research-to-Practice Conference in Adult, Continuing, and Community Education, The Ohio State University, Columbus, OH, p. 82-88.

Gouvêa, M. C. M. e Nakamoto, P. T. (2015). Avaliação de software educacional: uma oportunidade de reflexão da educação na sociedade do conhecimento. Encontro de Pesquisa Em Educação (VIII).

Haguenauer, C., Carvalho, F., Victorino, A. L., Lopes, M. e Cordeiro Filho, F. (2007). Uso de Jogos na Educação Online: a Experiência do LATEC/UFRJ, In: Revista EducaOnline, v.1, n.1, p.1–14.

Krasner, G. and Pope, S. (1988). A description of the model-view-controller user interface paradigm in the smalltalk80 system. In: Journal of Object-oriented Programming – JOOP.

Krone, C. (2013). Validação de Heurísticas de Usabilidade para Celulares Touchscreen*,* http://www.gqs.ufsc.br/wp-content/uploads/2013/07/WorkingPaper_WP_GQS_01-2013_v10.pdf.

Landis, J and Koch, G. (1977). "The Measurement of Observer Agreement for Categorical Data", In: Biometrics. p. 159-174.

Levin, J. e Fox, J. A. (2004). "Estatística para Ciências Humanas", São Paulo: Prentice Hall, 9 ed.

Likert, R. (1932). A technique for the measurement of attitudes. In: Archives of Psychology, n. 140, p. 44-53.

Lopes, A. (2012). "Desenvolvimento de um Jogo Didático para Ensino e Programação Orientada a Objetos e sua Aplicação em Cursos Técnicos de Computação", In: UFERSA – UERN, Mossoró - Brasil.

Medeiros, T. J. (2014). "Um Framework para Criação de Jogos Voltados para o Ensino de Lógica de Programação", Dissertação (Mestrado em Sistemas e Computação) - Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 80f.

Melo, V. K. S. L., Almeida, R. G., D'Emery, R.A. e Félix, Z. C. (2014). Desenvolvimento de um Jogo Educacional para auxiliar o ensino-aprendizagem de Introdução à Programação. In: Proceedings of XIX Conferência Internacional sobre Informática na Educação (TISE), Fortaleza, p.615-618.

Monclar, R., Silva, M. A. e Xexéo, G. (2018). Jogos com Propósito para o Ensino de Programação. In: Proceedings of SBGames 2018, Seção 2, 9., http://www.sbgames.org/sbgames2018/files/papers/EducacaoFull/188132.pdf

Nilsen, J. (1993). Usability Engineering. EUA: Morgan Kaufmann.

Nielsen, J. (1994). "10 Usability Heuristics for User Interface Design", https://www.nngroup.com/articles/ten-usability-heuristics/, Dezembro.

Oliveira, M., Souza, A., Ferreira A. e Barreiros, E. (2014). Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência, In: Anais do XXII Workshop sobre Educação em Computação - SBC, pp. 239-248.

Pereira, J. e Rapkiewicz, C. (2004). "O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma Visão Crítica da Pesquisa no Brasil", WEI RJES.

Plackett, R. L. (1983). Karl Pearson and the Chi-Squared Test. In: International Statistical Review. v. 51, n. 1, p. 59-72.

Pygame. (2020). Pygame SDL Library, https://www.pygame.org/news

Python. (2020). Site Oficial, https://docs.python.org/3/

Ramos, N., Freitas C., Avila, S., Costa, P. D., Testoni V. e Borin, J. F. (2015). Ensino de Programação para Alunas de Ensino Médio: Relato de uma Experiência. In: XXIII Anais do Workshop sobre Educação em Computação.

Reategui, E. (2007). Interfaces para Softwares Educativos. In: Renote – Revista Novas Tecnologias na Educação, v. 5, n. 1. 10 p.

Rocha, P. S., Ferreira B., Monteiro, B., Nunes, D. S. C. e Góes, H. C. N. (2015). Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. Revista Renote*, v.*8, n.3.

Santos, C., Silva, J. e Genz, C. (2017). Lógica de Programação: Iniciação Lúdica com Play Code Dog. In: Anais do WIE-CBIE 2017, v.23, n.1, p.108-117.

Scherer, N. P. (2018). Avaliação Heurística e Teste de Usabilidade para Softwares de Design de Interiores", Monografia (Ciência da Computação) – a Universidade Tecnológica Federal do Paraná, 53 p.

Shoukry, L., Sturm, C. Galal-Edeen, G. H. (2014). Pre-MEGa: A Proposed Framework for the Design and Evaluation of Preschoolers' Mobile Educational Games. In: Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering, v.313, n.1, p. 385-390.

Siegel, S. (1975). Estatística Não-paramétrica Para as Ciências do Comportamento. São Paulo: McGraw-Hill.

Silva, J. E. N. e D'Emery, R. A. (2021). Avaliação de Usabilidade do Software Educacional PaintCode, Relatório Técnico – Universidade Federal Rural de Pernambuco, Serra Talhada, Fevereiro, 21 p.

Silva, L. S. e Cavalcanti, E R. (2018). Avaliação Experimental do Robomind no Ensino de Programação com Estudantes do Curso Técnico em Informática Integrado ao Ensino Médio. In: Anais do VII SBIE-CBIE, v. 7, n.1., p. 288.

Silva, S. (2003). Critérios da Usabilidade: Um auxílio à qualidade do software. In: Revista Vértices, v. 5, n. 2, p.111-122.

Valle, P., Vilela, R., Júnior, P., Inocêncio, A. (2013) HEDEG - Heurísticas para Avaliação de Jogos Educacionais Digitais. In: XVIII Conferência Internacional sobre Informática na Educação. p.9-11.

Vieira, S. (2006). "Análise de variância: ANOVA". São Paulo, Atlas.

WHO. (2021). "WHO Coronavirus Disease (COVID-19) Dashboard", World Health Organization, https://covid19.who.int/, Fevereiro.