AlgoMixer: Explorando Designs para Interface Tangível de Algoritmos Sonoros

Cassiano Viana, André Raabe

Mestrado em Computação Aplicada
Laboratório de Inovação Tecnológica na Educação - LITE
Universidade do Vale do Itajaí (Univali)
Itajaí - SC - Brasil

cassiano.viana@edu.univali.br, raabe@univali.br

Abstract. *Tangible programming interfaces can assist in learning abstract concepts of programming logic as well as favoring the inclusion of specific groups such as the visually impaired. This paper presents a tangible interface tool for creating sound algorithms designed to regular users and to visually impaired users. The text details the process of developing the first version of the interface, the evaluations performed and the problems encountered. It describes the design of a new version (called AlgoMixer) that seeks to solve the problems encountered.*

Resumo. *Interfaces tangíveis de programação podem auxiliar o aprendizado de conceitos abstratos da lógica de programação e também favorecer inclusão de grupos específicos como os deficientes visuais. Este artigo apresenta uma ferramenta com interface tangível para criação de algoritmos sonoros projetada para atender usuários videntes e deficientes visuais. O texto detalha o processo de desenvolvimento da primeira versão da interface, as avaliações realizadas e os problemas encontrados. Descreve ainda o projeto de uma nova versão (denominada AlgoMixer) que busca solucionar os problemas encontrados.*

1. Introdução

A popularização do Pensamento computacional tem impulsionado, nos últimos anos, a busca de formas para estimulá-lo nas crianças. Uma vez que esses indivíduos terão (e já tem) uma vida influenciada pela computação, elas devem começar a trabalhar com algoritmos e solução de problemas já na educação infantil e fundamental [Barr e Stephenson 2011]. A definição de Pensamento Computacional recebeu várias contribuições desde que o termo foi cunhado por Wing em 2006. No entanto, entre todas as contribuições, é comum encontrar referências à capacidade de abstração, decomposição, generalização, pensamento paralelo e pensamento algorítmico [Selby et al. 2010].

Uma das formas de estimular essas habilidades é por meio de artefatos cujo propósito ou forma de uso tenham sido pensados para esse fim. Nesse contexto, considerar pessoas com deficiência é uma questão ética que envolve desafios de implementação e design. Especificamente no caso das pessoas com deficiência visual[1], o uso de informações exclusivamente visuais suprime alunos que poderiam aprender tais conteúdos [Kakehashi et al. 2013].

Visando incluir esse público, foi desenvolvida uma interface de programação tangível com metáfora sonora. Ela permite aplicar conceitos de algoritmo como sequenciamento,

[1] Dados de 2013 revelam que no Brasil há mais de 7,5 milhões de pessoas com deficiência [IBGE 2013]

DOI: 10.5753/cbie.wcbie.2019.465

repetição e desvio para tocar música em uma bateria programável. A Figura 1 mostra a interface em uso.

Figura 1: Interface de algoritmos sonoros desenvolvida

O objetivo é permitir, que tanto deficientes visuais como outros grupos possam compor sequências sonoras ao mesmo tempo que assimilam conceitos de algoritmos como repetição e desvio. O funcionamento se dá pela leitura de blocos de madeira, que são capturados pela câmera de um celular. Através do posicionamento, os sons associados a cada peça do conjunto são tocados na sequência da esquerda para direita, podendo sofrer modificações. Dentre essas modificações, a repetição ou o condicionamento abarcam os conceitos de programação básicos citados. Em relação à acessibilidade, cada peça possui inscrições em Braille. Além disso, instruções sonoras presentes em um modo específico de ajuda servem de apoio ao entendimento da função de cada bloco.

No entanto, resultados qualitativos de avaliações prévias de usabilidade mostraram que a utilização de inscrições em Braille não foi suficiente para atender de forma plena o público dos deficientes visuais [Viana e Raabe 2018]. Isso por dois motivos: (1) nem todos os cegos dominam o alfabeto Braille e (2) sem uma indicação correta da rotação da peça, o usuário não consegue iniciar rapidamente a leitura.

Como a interface não foi pensada unicamente para o público dos deficientes visuais, foram observados padrões de comportamento singulares também nas pessoas que enxergam. Esses padrões dizem respeito a erros de leitura e significação das peças por características de desenho. A interface possui blocos com nomes CAIXA, PRATO e SURDO. Embora sejam também peças de bateria, essas palavras podem se encaixar em mais de um contexto.

O artigo, antes discutir as questões de usabilidade apresenta o conceito de TPI (*Tangible Programming Interfaces*), que são interfaces tangíveis voltadas para programação na seção 2. Também detalha o funcionamento da AlgoMixer, que é a interface de algoritmos sonoros desenvolvida na seção 3. A Seção 4 apresenta a metodologia aplicada na identificação de *issues*[2]. A Seção 5 apresenta as *issues* e respectivas propostas de soluções. A Seção 6 discute sobre as avaliações e compara as dimensões cognitivas de notações aplicadas à interface avaliada. A Seção 7 apresenta as conclusões e trabalhos futuros.

2. Interfaces de Programação Tangível

A busca por formas e interfaces para programar o computador inicia junto com a construção das primeiras máquinas. Foi Seymour Papert quem trouxe o tema para o âmbito educacional

[2] O termo *issues* foi escolhido porque é comum para identificar bugs, problemas e sugestões em projetos de software.

ao projetar com colegas a linguagem Logo em que crianças usavam o computador para programar uma tartaruga robótica [Horn e Bers 2019]. Nas Interfaces de programação tangíveis o controle dos comandos do programa passa literalmente para as mãos do usuário em um conceito da manipulação direta dos "bits" [Ishii 2008]. Enquanto que na interface gráfica esse controle se dá por do teclado ou mouse na tela, nas interfaces tangíveis a informação é representada diretamente pelos objetos. Embora teclado e mouse também sejam objetos tangíveis, eles não representam a informação em questão e são somente meio de acesso, criação e transformação. No universo infantil, a manipulação de objetos em forma de brinquedos é inerente às interfaces tangíveis, favorecendo sua aplicação no apoio ao ensino de algoritmos.

Para citar alguns exemplos clássicos dessas interfaces, temos o AlgoBlock (Figura 1A), que é formado por um conjunto de blocos eletrônicos que, quando interligados, controlam um submarino na tela [Suzuki e Kato [S.d.]]; o Tern (Figura 1B) é uma linguagem de programação que usa marcas fiduciais e tem como fim controlar um robô em um museu [Horn e Jacob 2007]; os Tangible Programming Bricks (Figura 1C), que são mini-blocos eletrônicos encaixáveis para formar um programa e; A P-CUBE (Figura 1D) que é uma linguagem de programação acessível para os deficientes visuais.

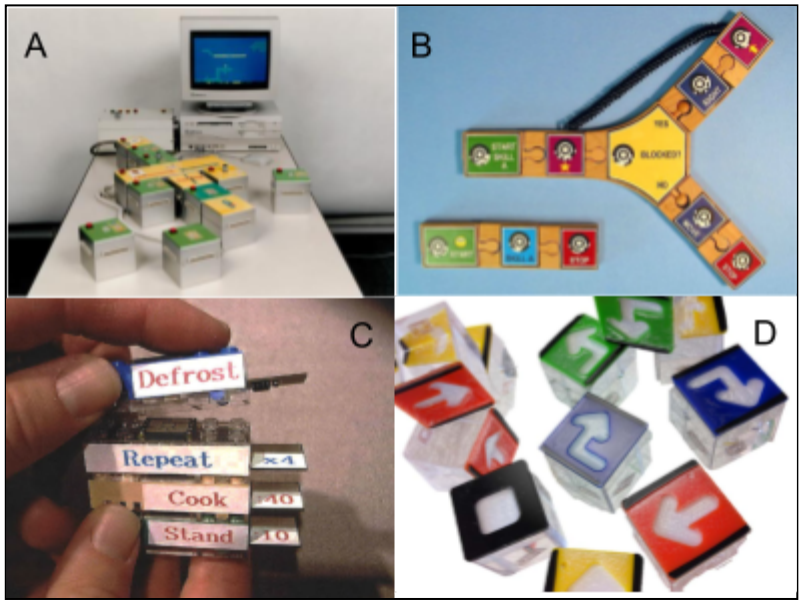


Figura 2: Interfaces de programação tangíveis Algoblock (a), Tern (b), Tangible Programming Bricks (c) e P-CUBE (d)

A forma da de organizar, encaixar, aproximar e mover os elementos da interface são geralmente próprios de cada linguagem. Além disso, a relação das peças com o ambiente e os retornos também são inerentes à tecnologia utilizada. Enquanto Tern e Tangible Bricks utilizam componentes eletrônicos em todos os elementos, Tern identifica os blocos com o uso de marcas fiduciais e P-CUBE lê com RFID (Radio-frequency identification) as peças colocadas em uma mesa de encaixe.

Contudo, é perfeitamente possível utilizar interfaces tangíveis para construir estruturas abstratas de informação. As chamadas dimensões cognitivas de notações, geralmente utilizadas para análise de interfaces gráficas, foram abstraídas para estudar interfaces tangíveis [Edge e Blackwell 2006]. Dentre essas dimensões, a ***viscosidade*** é especialmente importante nas interfaces tangíveis. Ela se refere a quão rígidas as ligações

entre os elementos são. No caso da interface AlgoMixer, os blocos de madeira possuem encaixes que são pouco restritivos, ou seja, possui baixa viscosidade. Isso é bom por ampliar as possibilidades de encaixe, mas ao mesmo tempo permite combinações inválidas. Outros elementos como visibilidade, dependências escondidas, e engajamento prematuro são dimensões cognitivas que fazem parte do conjunto de dimensões descritas por Green e Blackwell (1998) e que serão analisadas em relação a AlgoMixer. Mas antes é necessário entender seu propósito e funcionamento com mais detalhes.

3. Funcionamento da interface AlgoMixer

A AlgoMixer utiliza a metáfora de uma bateria. O conceito principal é de uma linha temporal que se move da esquerda para direita tocando as peças de bateria. Sobre as peças musicais são posicionados blocos de programação, que permitem a variação do som conforme condição ou repetição. A Figura 3 ilustra a base da interface mostrando os tipos de peças (não são as peças finais, somente a representação abstrata das peças de controle parâmetro e som).

Figura 3: base da interface AlgoMixer, com tipos genéricos das peças e funcionamento da linha de tempo.

Nas partes inferior esquerda e direita respectivamente há botões giratórios fixos para controle de velocidade e volume global. Ainda, o volume individual pode ser controlado pela posição vertical da peça em cada uma das oito casas disponíveis. Na parte central inferior podem ser encaixadas peças especiais para controle de condição. Essas peças são a estrela e o círculo e podem ser feitas condicionais do tipo: “se a estrela estiver presente no tabuleiro (encaixada em Figura 3, indicação A) então toca o bumbo”[3]. Já na instrução de repetição, pode-se formar instruções do tipo “repita 3 vezes a caixa”, utilizando a peça repita em conjunto com peças de parâmetro. A Figura 4 ilustra as instruções de controle SE e REPITA. As forma de uso dos parâmetros está apresentada na Figura 4, indicações A e B. Eles ficam encaixados na parte superior da interface, entre a instrução de controle, e a peça de som afetada.

[3] Nesse caso o bumbo é um exemplo de peça de som, representados pelas peças amarelas.

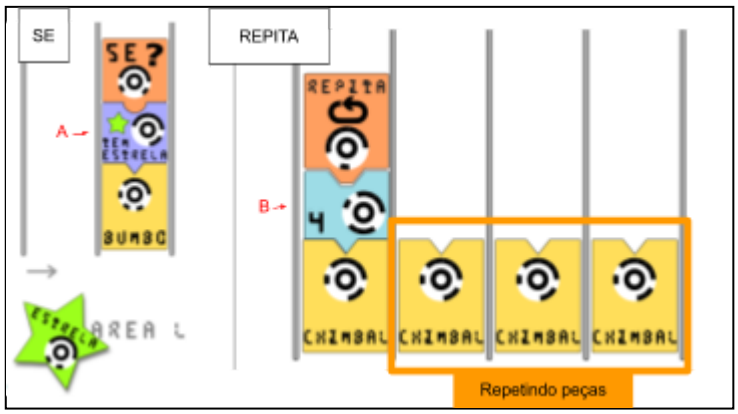


Figura 4: Funcionamento das peças SE e REPITA.

No caso ilustrado da repetição, o efeito de colocar a peça repita com o parâmetro "4" sobre o chimbal é replicar a peça amarela "virtualmente" mais três vezes para a direita. No entanto, observou-se com frequência o uso desse padrão nas últimas casas. Isso foi um problema porque quando o número de repetições excede o número de casas o comportamento do som é indefinido.

Dentro dos tipos de blocos, ainda há um bloco de ativação de ações. TOCAR, PARAR, AJUDA e GRAVAÇÃO permitem controlar o iniciar e parar da música, ouvir instruções de ajuda e gravar sons externos em peças especiais em branco. A funcionalidade de gravação foi de longe a mais engajante. Permitir compor uma música com a própria voz, palmas ou assobio caracteriza a "música" com os sons improvisados pelo usuário.

Em relação à tecnologia, foi utilizada a biblioteca de marcas fiduciais TopCodes para reconhecimento das peças. Ela permite obter posicionamento, giro e diâmetro das peças [Lab e Horn 2019]. A implementação escolhida foi para a plataforma Android[4], por evitar a necessidade do desenvolvimento de *hardware* específico na perspectiva de se desenvolver uma interface de baixo custo.

4. Metodologia de descoberta de issues

A metodologia escolhida para a descoberta de questões relativas à usabilidade e erros foi dividida em duas partes: uso guiado e uso livre. O uso guiado foi aplicado para um participante apenas com deficiência visual (Figura 5A). O uso livre foi aplicado em uma sala de aula com alunos sem ajuda e sem deficiência visual (Figura 5B). As perguntas e as dificuldades apresentadas foram anotadas para composição da lista de *issues*.

[4] O aplicativo pode ser baixado em https://hd1.nyc3.cdn.digitaloceanspaces.com/ttc/SoundProgrammingDrum.apk

Figura 5: Uso individual guiado (esquerda) e uso livre em grupo (direita).

4.1. Uso guiado

O participante do uso guiado foi um usuário é adulto, possui experiência prévia com programação, mas não com interfaces tangíveis para o fim de programação. Foi apresentado à interface, e requisitado que realizasse três tarefas.

1. Sequenciamento: Colocar algumas peças sonoras sobre a mesa e identificar cada som.
2. Repetição: Unir blocos de repetição, parâmetro e som para multiplicar o efeito de um som
3. Desvio condicional: Unir blocos de condição, parâmetro e som para executar uma determinada batida ou não.

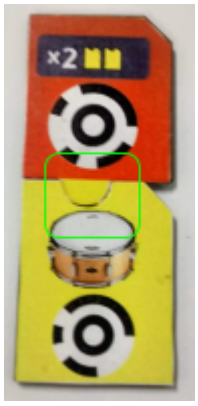
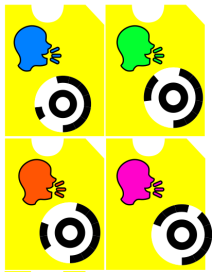
Todas as tarefas foram primeiramente mostradas ao usuário, depois requisitado a ele que as reproduzisse. Foram utilizadas apenas as peças necessárias para cada atividade para que o usuário não precisasse procurar entre as várias peças (cerca de 30 peças, tornou-se difícil até mesmo para quem enxerga encontrá-las). O usuário afirmou compreender as três atividades quando explicadas praticamente.

4.2 Uso livre

O uso livre foi praticado durante uma aula de Pensamento Computacional no laboratório de educação da universidade com 8 alunos de 7 a 8 anos que possuíam experiência com programação em computador. O pesquisador não solicitou que os participantes utilizassem o brinquedo, nem aos pais autorização formal para uso. A participação ocorreu de forma espontânea durante uma aula de 50 minutos e não foram solicitadas tarefas específicas. O objetivo foi analisar como os participantes usariam a interface sem explicação prévia, oferecendo apenas respostas pontuais. Nesse caso, a interface foi utilizada por até três alunos de uma única vez. Houve interação com o pesquisador, pois este estava posicionado ao lado do brinquedo e consequentemente dos participantes.

5. Issues e soluções

A cada nova versão a interface de programação sonora recebe ajustes realizados através do design centrado no usuário (DCU). Esse é um termo bastante aberto usado para descrever processos de desenvolvimento em que os usuários finais influenciam em como o produto ganha forma [Abras et al. 2004]. No presente está sendo desenvolvida uma nova versão que deve resolver as questões apresentadas no Quadro 1. Essas questões foram levantadas através das duas avaliações feitas com uso guiado e livre.

Issue	Ocorrência	Explicação	Solução	Proposta
Encaixe incorreto da peça SE com parâmetros de repetição		Embora os encaixes entre as peças SE e "4" sejam incompatíveis, elas "quase" encaixam. Assim o usuário acaba realizando o erro sintático.	Criar um encaixe único para as peças de controle (**SE** e **REPITA**) e som removendo a necessidade do parâmetro.	 (repita 2x a caixa)
Usuários perdem a peça gravada.		Como as peças de gravação são todas iguais, não é possível associar cada som à peça visualmente. Embora as peças possam receber desenhos com canetinha para identificação, isso quase nunca é feito.	Criar peças com identificadores coloridos diferentes para cada peça de gravação. Essa solução aumenta a ***visibilidade*** dos elementos.	 (4 peças que permitem gravar 4 sons diferentes externos - um para cada cor)
Usuários não associam as peças aos sons de bateria.		Nomes genéricos como CAIXA e PRATO tem dois significados.	Adicionar imagens das peças de bateria tende a ajudar os usuários entender que se trata de uma bateria. Prato, caixa bumbo, surdo, chimbal são peças de bateria)	 (pela com desenho da caixa de bateria)

Usuário não entende como aproximar a peça da câmera para gravá-la com um som externo.		O movimento de aproximação da peça do celular ocorre longe do tabuleiro, o que é contraintuitivo.	Criar uma área na base específica para gravação. Essa área ficará sempre visível com um acesso mais rápido, sem a necessidade de ativar a função GRAVAÇÃO.	 (área de gravação)

Quadro 1. *Issues* e soluções propostas

Durante o uso com o deficiente visual percebeu-se que o tatear sobre as peças bloqueia a visão da câmera. Por isso pretende-se buscar uma forma de interação que não dependa desse tipo de organização.

Nessa versão específica o texto em braille foi removido. Durante o uso com o deficiente visual isso não pode ser avaliado já que ele não dominava esse sistema de escrita. No entanto será devolvida na próxima versão com impressão 3D. Para isso será utilizado o software Braille 3D, disponível no site do FabLab POALAB[5].

6. Discussão

Tanto no uso guiado como no uso livre, os participantes demonstraram compreender as funções da interface. O deficiente visual teve um uso mais metódico e utilizou menos peças, no máximo 6 (3 de som) simultâneas, o que facilitou a identificação. As crianças do segundo teste utilizaram muitas peças. “Muitas” significa preencher toda a parte superior com peças a ponto de ficar praticamente impossível compreender os efeitos dos fluxos de repetição e desvio gerados. Mesmo assim, o “ruído” gerado engajou os participantes, principalmente quando associados à função de gravação de sons externos (gravaram a própria voz e palmas). Porém essa forma de uso (muitas peças com pouca organização) é um problema porque não reforça a aplicação analítica, detalhada de algoritmos. Dessa forma, os conceitos de algoritmos trabalhados (repetição e desvio) foram ofuscados pela experimentação da criação de sons.

Especificamente em relação às variáveis de dimensões cognitivas, considera-se a **viscosidade**, que é a relação de restrição e liberdade de encaixes entre os elementos, deve ser reforçada, pois os encaixes livres permitiram usos incorretos. As **dependências escondidas**, foram observadas entre as peças de parâmetro de condição. Mais claramente: a peça “se tem estrela”, depende da peça estrela, ou seja, as duas peças devem estar presentes juntas para o efeito ocorrer. Essa relação escondida não foi observada por nenhum dos participantes da pesquisa. O **engajamento prematuro** foi observado pelos usuários do uso livre. O engajamento prematuro é o uso apressado dos elementos de forma incorreta sem a devida análise. Isso foi observado durante o uso livre. A **visibilidade** é uma dimensão cognitiva de uso em que as informações são facilmente visíveis. Isso não foi observado na peça de ações. Por ser uma espécie de dado, pelo menos uma função fica escondida na face que fica para baixo.

[5] https://www.poalab.net.br/t2b/

7. Conclusão

Utilizar como base as dimensões cognitivas de notações facilitou observar o relacionamento entre os componentes da interface sob diferentes perspectivas. As alterações na interface com vista a produzir a interface AlgoMixer consideram essas dimensões associadas às necessidades do público com visão e sem visão.

As avaliações realizadas com uso livre e guiado auxiliaram na descoberta das questões a serem resolvidas, mas ainda há carência no número de participantes deficientes visuais. Isso deve ser resolvido com a participação de uma instituição de apoio aos deficientes visuais da região, com a passagem pela aprovação do projeto em comitê de ética.

Como trabalho futuro, serão implementadas as alterações descritas para essa avaliação. Em seguida serão realizados testes de usabilidade com foco no público dos deficientes visuais e não visuais para comparar a adequação da nova versão a ambos.

8. Referências

Abras, C., Maloney-Krichmar, D. and Preece, J. (2004). User-Centered Design. p. 14.

Barr, V. and Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Transactions on Computational Logic*, p. 12.

Edge, D. and Blackwell, A. (aug 2006). Correlates of the cognitive dimensions for tangible user interface. *Journal of Visual Languages & Computing*, v. 17, n. 4, p. 366–394.

Green, T. and Blackwell, A. (1998). Cognitive Dimensions of Information Artefacts: a tutorial.

Horn, M. and Bers, M. (2019). Tangible Computing. In: Fincher, S. A.; Robins, A. V.[Eds.]. . *The Cambridge Handbook of Computing Education Research*. 1. ed. Cambridge University Press. p. 663–678.

Horn, M. S. and Jacob, R. J. K. (2007). Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*. . ACM Press. http://portal.acm.org/citation.cfm?doid=1226969.1227003, [accessed on Sep 25].

IBGE, I. (2013). Tabela 5754: Pessoas com deficiência visual, total, percentual e coeficiente de variação, por nível de instrução e situação do domicílio. https://sidra.ibge.gov.br/tabela/5754#resultado, [accessed on Sep 25].

Ishii, H. (2008). Tangible bits: beyond pixels. In *Proceedings of the 2nd international conference on Tangible and embedded interaction - TEI '08*. . ACM Press. http://portal.acm.org/citation.cfm?doid=1347390.1347392, [accessed on Sep 25].

Kakehashi, S., Motoyoshi, T., Koyanagi, K., Ohshima, T. and Kawakami, H. (dec 2013). P-CUBE: Block Type Programming Tool for Visual Impairments. In *2013 Conference on Technologies and Applications of Artificial Intelligence*. . IEEE. http://ieeexplore.ieee.org/document/6783884/, [accessed on Jun 15].

Lab, T. and Horn, M. (2019). *TIDAL-Lab/TopCodes*.

Selby, C. C., Selby, C., Woollard, John and Woollard, J (2010). Computational Thinking: The Developing Definition. *2010*, p. 6.

Suzuki, H. and Kato, H. ([S.d.]). AlgoBlock: a Tangible Programming Language - a Tool for Collaborative Learning.

Viana, C. and Raabe, A. (28 oct 2018). Interface de programação tangível para produção de algoritmos sonoros. . http://br-ie.org/pub/index.php/wcbie/article/view/8219, [accessed on Jun 17].