UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Ana Luísa Veroneze Solórzano

UMA PLATAFORMA DE JOGOS PARA INTRODUÇÃO À PROGRAMAÇÃO PARALELA UTILIZANDO LINGUAGEM VISUAL BASEADA EM BLOCOS

Santa Maria, RS
2019

Ana Luísa Veroneze Solórzano

UMA PLATAFORMA DE JOGOS PARA INTRODUÇÃO À PROGRAMAÇÃO PARALELA UTILIZANDO LINGUAGEM VISUAL BASEADA EM BLOCOS

Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADORA: Prof.ª Andrea Schwertner Charão

459
Santa Maria, RS
2019

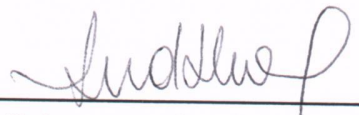
End. Eletr.: alsolorzano@inf.ufsm.br

Ana Luisa Veroneze Solórzano

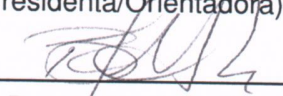
UMA PLATAFORMA DE JOGOS PARA INTRODUÇÃO À PROGRAMAÇÃO PARALELA UTILIZANDO LINGUAGEM VISUAL BASEADA EM BLOCOS

Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

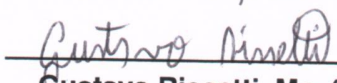
Aprovado em 2 de dezembro de 2019:



Andrea Schwertner Charão, Dra. (UFSM)
(Presidenta/Orientadora)



Roseclea Duarte Medina, Dra. (UFSM)



Gustavo Rissetti, Ms. (IFFarroupilha)

Santa Maria, RS
2019

AGRADECIMENTOS

Agradeço primeiramente à minha orientadora, minha maior inspiração desde o primeiro ano de curso, por todos os ensinamentos e conversas em sua sala, que me faziam sair motivada e com várias ideias. Obrigada por fortalecer minha inquietude mostrando que a Universidade vai além da sala de aula.

Agradeço à minha família, pelo suporte aos meus estudos em Santa Maria, em especial à minha irmã, que conviveu comigo durante grande parte desta jornada. Ao BitMarias, que me trouxe amigas incríveis da computação, que tornaram as idas à UFSM ainda mais especiais. Aos meus amigos de Santa Maria, em especial meus colegas de curso, que me apoiaram durante a graduação, e aos meus amigos de Porto Alegre, Priscila e Wagner, que mesmo longe fizeram o possível para manter a amizade e o incentivo.

If you can't do great things, do small things in a great way.

(Napoleon Hill)

RESUMO

UMA PLATAFORMA DE JOGOS PARA INTRODUÇÃO À PROGRAMAÇÃO PARALELA UTILIZANDO LINGUAGEM VISUAL BASEADA EM BLOCOS

AUTORA: Ana Luísa Veroneze Solórzano
ORIENTADORA: Andrea Schwertner Charão

Devido aos avanços das tecnologias digitais, sistemas *multicore* estão presentes em notebooks, desktops e smartphones. Para obter melhor desempenho nesses dispositivos, podem ser utilizadas abordagens de programação paralela. Em muitos cursos de computação, disciplinas de programação paralela são ofertadas após disciplinas de introdução à programação. Com isso, o aluno é induzido a desenvolver programas sequenciais e a implementar paralelismo em seu código apenas se houver posterior necessidade de otimização. Entretanto, esse fluxo nem sempre é uma maneira eficaz de criar programas eficientes em comparação ao desenvolvimento inicial de programas paralelos. Linguagens baseadas em blocos vêm se popularizando como uma abordagem para introdução à programação. Normalmente, essas ferramentas são apresentadas no formato de jogos e tutoriais, com estruturas sequenciais e programação orientada a eventos. Este trabalho apresenta BlocklyPar, um conjunto de três jogos para o ensino-aprendizagem de programação paralela utilizando programação visual com blocos, voltado a calouros de cursos superiores de Computação. Cada jogo apresenta fases com níveis de dificuldade que aumentam gradualmente, e cenários com contextos comuns ao dia-a-dia do público-alvo, como ir à biblioteca ou à sala de aula. Através dos testes realizados notou-se que o BlocklyPar tem potencial para instigar o pensamento paralelo em alunos de ensino superior com diferentes níveis de conhecimento em programação, para que utilizem abordagens paralelas naturalmente em suas aplicações. A plataforma está disponível para acesso online e seu código está hospedado em um repositório público para continuidade do projeto.

Palavras-chave: Programação paralela. Pensamento paralelo. Programação com blocos. Linguagem baseada em blocos. Jogo educacional. BlocklyPar.

ABSTRACT

A GAMES PLATFORM FOR INTRODUCING PARALLEL PROGRAMMING WITH BLOCK-BASED LANGUAGE

AUTHOR: Ana Luísa Veroneze Solórzano
ADVISOR: Andrea Schwertner Charão

Due to the advances in technology, multicore systems are found in notebooks, desktops, and smartphones nowadays. With parallel programming approaches inserted in programming codes, we can harness the computing power on these devices. In many computing majors, parallel programming subjects are usually offered as an optional class, after basic programming classes. With this, the students first develop knowledge about sequential programming, not considering the resources of the computational environment used, and then they are challenged to implement parallelism in their codes if there is further need for optimization. However, this can be an inefficient approach to develop high-performance programs, compared to start the development considering parallel approaches. Block-based languages have become a popular alternative to introducing coding to kids and learners in Computer Science. Tools that use block-based programming usually offer games and tutorials activities, using sequential data structures and event-oriented programming. This project presents BlocklyPar, a set of three tutorials for teaching parallel programming using block-based programming, target to freshmen students from higher education courses in Computer Science. The difficulty levels of the games increase linearly, and each game has scenarios involving student's day-to-day tasks, such as going to the library and to the classroom. With the user tests, we observed that BlocklyPar has potential to instigates parallel thinking in higher education students with different levels of knowledge about programming so that they can use parallel approaches naturally in their applications. The platform is available online and its code is hosted on a public repository to promote the continuity of this project.

Keywords: Parallel programming. Parallel thinking. Programming with blocks. Block-based language. Educational game. BlocklyPar.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Arquiteturas multicore são uma realidade presente no nosso dia-a-dia, tanto em smartphones como em supercomputadores que executam programas de alto desempenho. Atualmente, existem demandas crescentes de maior poder computacional, principalmente com a popularização dos campos de Big Data e Deep Learning, aplicados em diversas áreas, como biologia, meteorologia e computação gráfica (GARDNER, 2017).

A programação paralela é a área da Ciência da Computação que proporciona recursos para a melhor utilização dos ambientes multicore, através de bibliotecas e API's aplicadas em linguagens de programação textual. Para isso, o programador deve saber escrever, compilar e depurar seus códigos utilizando ferramentas como Pthreads (IEEE, 2016), OpenMP (CHAPMAN; JOST; PAS, 2007) e MPI (PACHECO, 1996).

Normalmente, disciplinas de programação paralela são ofertadas como optativas nos cursos de computação, após disciplinas básicas de programação, e utilizam livros e práticas em sala de aula como recursos didáticos. Nesse contexto, os alunos têm acesso às tecnologias e aos recursos, mas não são instigados desde o início do curso a pensar paralelamente e adquirir a ideia de processadores executando instruções ao mesmo tempo.

Para o aluno, adquirir autonomia com programação sequencial e depois retroceder a conceitos básicos de programação, servindo de base para pensar em como introduzir o paralelismo, não é uma maneira eficaz de desenvolver programas eficientes. Inclusive, é citado como lamentável que os programadores deixem de utilizar todo o poder computacional disponível porque não têm treinamento suficiente para isso (GARDNER, 2017).

A programação paralela não precisa ser difícil de ensinar ou aprender. O aluno pode contextualizar a necessidade de explorar computadores *multicore* a aplicações reais que processam grandes quantidades de dados em áreas como meteorologia, agricultura e saúde. O desafio está em quais ferramentas utilizar nesse processo, visto que o ensino-aprendizagem de programação, sem introdução ao paralelismo, já é visto como um desafio.

1.1 CONTEXTO

Programar computadores é considerada uma habilidade difícil de ser aprendida (ROY; ROUSSE; DEMERITT, 2012). Iniciantes em programação, precisam aprender sintaxes novas que devem ser seguidas com exatidão para correta execução de seus programas (KELLEHER; PAUSCH, 2005). Também, alunos com algum conhecimento prévio de programação podem enfrentar dificuldades para compreender conceitos avançados durante um curso superior (SHUHIDAN; HAMILTON; D'SOUZA, 2011).

Linguagens de programação baseadas em blocos são alternativas para o ensino-aprendizagem de programação de computadores. Os blocos são representações visuais de códigos de programas que adicionam à sintaxe usual de linguagens textuais, componentes com formas e cores distintos, que realizam ações quando combinados através do simples arraste e encaixe de blocos (SHAPIRO; AHRENS, 2016).

A vantagem deste recurso está em se libertar de tarefas complexas comuns a algumas linguagens de programação textual, por exemplo, verificação de sintaxe, dependências de dados e alocações de memória. Esta implementação deixa o programador menos exposto a erros, pois deve associar e assimilar blocos organizados por conceitos, como blocos textuais, matemáticos e condicionais, por cores, e por formas, que podem ou não se encaixar. Além disso, simplificam a compreensão de estruturas de dados, que são abstraídas em um único bloco tratado como uma entidade.

Universidades internacionalmente reconhecidas utilizam ferramentas de programação com blocos como introdutórias à programação textual. Em cursos de Introdução à Ciência da Computação, a Universidade de Harvard realiza atividades com Scratch antes de ensinar C, e a Universidade de Berkeley utiliza Snap! antes de ensinar Python (BAU et al., 2017). Em escolas, alunos entre 6° e 8° ano que tiveram contato com uma ferramenta de programação com blocos antes de programação textual, alcançaram níveis cognitivos mais elevados para a compreensão da maioria dos conceitos de programação apresentados, em comparação aos alunos que não tiveram contato com a ferramenta (ARMONI; MEERBAUM-SALANT; BEN-ARI, 2015).

De acordo com PAPERT (1972), as crianças tendem a aprender mais ao realizar atividades práticas, refletindo sobre os comandos que irão utilizar e o impacto dos comandos em suas ações. A linguagem Logo[1], surgiu como um recurso de iniciação à programação, baseado na teoria do Construcionismo defendida por Papert, onde o usuário tem controle sobre as ações de um personagem, e trabalha construindo os comandos que designará ao personagem.

Existem diversas ferramentas de programação com blocos que, em sua maioria, baseiam-se em preceitos construcionistas, onde o usuário pode criar livremente animações, aplicativos de celular, ou respostas para desafios (WEINTROP; WILENSKY, 2015). Uma das mais conhecidas é a Hora do Código[2], uma iniciativa mundial para introdução à programação através de tutoriais com temas e desafios distintos. Acredita-se que o uso de tutoriais e jogos seja uma abordagem eficaz para o ensino de programação, pois envolve o usuário na resolução dos desafios ao mesmo tempo em que trabalha gradativamente aspectos específicos de programação.

Os tutoriais da Hora do Código utilizam estruturas sequenciais para a programação de respostas para os desafios. Entretanto, os computadores atuais possuem arquiteturas

[1] https://el.media.mit.edu/logo-foundation/

[2] https://code.org

multicore, que permitem ao usuário explorar o ambiente computacional, implementando códigos que executam em paralelo. Técnicas de programação paralela são inseridas em linguagens de programação textual, voltadas principalmente ao aumento de desempenho de programas, diminuindo seu tempo de execução (SHAFI et al., 2014).

Utilizando programação paralela, o usuário pode processar grandes quantidades de dados em tempos muito menores do que se executasse serialmente, podendo aproveitar o tempo de execução poupado para realizar otimizações e novas implementações em seu código. Assim, além do pensamento computacional, para a programação sequencial de computadores, o pensamento paralelo pode ser considerado uma habilidade essencial aos programadores atualmente.

Disciplinas de programação paralela geralmente são ofertadas como disciplinas complementares de graduação. Com isso, os alunos são orientados a aprender programação sequencial e após, se desejarem, aprender programação paralela. Porém, essa ordem de aprendizado não precisaria ser seguida, pois o conhecimento e a prática de conceitos paralelos é tão importante quanto o conhecimento de conceitos lógicos básicos de programação, tais como laços e condicionais (FENG; GARDNER; FENG, 2017).

Existem materiais e referências para a implementação de programas paralelos, mas o aprendizado do pensamento paralelo não parece ser sanado completamente por esses recursos (MELLOR-CRUMMEY; GROPP; HERLIHY, 2010). Ou seja, os alunos se tornam aptos a otimizarem seus programas com abordagens paralelas, mas não criam seus algoritmos já considerando o potencial do ambiente *multicore* em que serão executados. Visto isso, supõe-se que ferramentas para o ensino de programação paralela, utilizando métodos que instigam nos alunos o pensamento paralelo de forma natural, seriam benéficas se usadas no ensino de programação desde o início.

1.2 OBJETIVOS

Este trabalho tem o objetivo de desenvolver uma plataforma para introduzir a programação paralela de maneira lúdica e atrativa para alunos de cursos superiores de Computação que estão aprendendo programação. A plataforma foi inspirada na Hora do Código e nos Jogos do Blockly[3], trazendo desafios que exercitem o pensamento paralelo e despertem o interesse na área. Para isso, utilizou-se uma linguagem baseada em blocos, visto o potencial desse recurso como introdutório à programação.

A seguir, são listados os objetivos específicos deste trabalho:

- Investigar ferramentas de programação com blocos existentes, analisando suas possibilidades e suas limitações para aplicação de paralelismo de forma eficiente, con-

[3] https://blockly-games.appspot.com/

siderando a linguagem de programação textual que é utilizada na implementação da ferramenta escolhida;

- Criar protótipos dos jogos, com fases de níveis de dificuldade distintos, que abordem diferentes conceitos de programação paralela, pensando tanto em recursos funcionais, como em recursos visuais, para torná-lo atrativo e compreensível aos usuários;
- Desenvolver os jogos utilizando a ferramenta escolhida, criando um recurso para introdução à programação paralela com linguagem de programação visual com blocos de comando;
- Implementar requisitos do projeto: (i) que tornem os jogos desafiador para os usuários, com aspectos comuns aos encontrados em jogos populares, (ii) que ofereçam informações sobre o meio em que os códigos paralelos criados estão sendo executados, mesmo que apresentando uma abstração frente ao ambiente real. Relacionar essa noção com conceitos de arquitetura de computadores, de modo com que o usuário esteja ciente de como está explorando a arquitetura utilizada (iii) que coletem informações sobre o acesso dos usuários, para pontuar melhorias e analisar o seu uso;
- Escolher um serviço que permita a hospedagem da plataforma em um ambiente de acesso web e realização do *deploy* do projeto em tal serviço, visando o fácil acesso dos usuários. Inserção de instruções e documentação necessária para compreensão da atividade;
- Realizar testes com usuários iniciantes em cursos superiores de Computação e em estágios avançados dos cursos, sem conhecimento de programação paralela, de forma a levantar melhorias para o projeto, sugestões e possíveis erros a serem corrigidos.

1.3 JUSTIFICATIVA

O uso de linguagens baseadas em blocos facilita o ensino de programação, oferecendo recursos úteis tanto para novatos como para programadores avançados. Um estudo realizado com 107 alunos ingressantes em Ciência da Computação, mostrou que os alunos que utilizaram programação com blocos antes de linguagem textual aumentaram o seu índice de desempenho acadêmico e tiveram maior retenção no curso, em comparação aos alunos que não utilizaram (MOSKAL; LURIE; COOPER, 2004).

Atualmente, a programação com blocos é aplicada como uma ferramenta para o desenvolvimento de habilidades de lógica e programação, sendo ainda necessário o

aprendizado de linguagens textuais para o desenvolvimento de programas de propósito geral (SHAPIRO; AHRENS, 2016). Entretanto, o aprendizado com blocos antes de abordar leitura e edição de programas textuais é compensado posteriormente, pois os alunos tendem a assimilar e compreender a programação textual com maior facilidade (WEINTROP, 2015).

Uma alternativa para encorajar alunos a praticarem programação é inserir habilidades específicas da área em jogos e tutoriais. Nos tutoriais da Hora do Código, por exemplo, o usuário realiza tarefas de repetição para ditar as ações de um personagem, enquanto avança no desafio auto-orientado (TUMLIN, 2017). Uma habilidade específica de computação para ser explorada nesse contexto é a programação paralela.

Visto que o uso de linguagens baseadas em blocos facilita a compreensão de programação, o uso de blocos para o ensino-aprendizagem de programação paralela é uma potencial abordagem para exercitar o pensamento paralelo. Assim, este trabalho buscou explorar este potencial, criando jogos que apoiem o aprendizado de programação paralela, onde o usuário criará soluções para os desafios utilizando blocos lógicos de comando.

ONTANON et al. (2017) apresentam o processo de design de um jogo para ensino de programação paralela, mas sem utilização de blocos. O jogo aborda conceitos básicos sobre utilização de paralelismo, como uso de *threads* e cálculo do *speedup*, diferenças entre programação serial e programação paralela, e a ideia de sincronismo de processos, tratando os conceitos de exclusão mútua, condição de corrida, *deadlock* e *starvation*.

Encontrou-se apenas um trabalho concluído que utiliza programação com blocos para o ensino de programação paralela. Seu foco é abordar conceitos de paralelismo, passando o paradigma do produtor-consumidor e a ideia de concorrência (FENG; GARDNER; FENG, 2017). Entretanto, não foram encontradas ferramentas para a prática do pensamento paralelo usando blocos que motivem os usuários a enxergarem seus problemas com potencial implementação de paralelismo, e a comparação da abordagem paralela à execução serial no desempenho de suas tarefas.

Acredita-se que não apenas a programação serial deva ser ensinada para alunos ingressantes no curso, mas também o pensamento paralelo. Supõe-se que a plataforma proposta neste trabalho possa apresentar o pensamento paralelo ao usuário de forma clara e intuitiva. Espera-se que ela incentive o público-alvo, iniciantes em programação, a perceberem gradualmente as diferenças entre o uso e o não uso de programação paralela em seus trabalhos, adquirindo a ideia de alocar recursos computacionais para realizar mais de uma tarefa ao mesmo tempo. Além disso, supõe-se que o uso de um tutorial, com início e fim, seja mais atraente aos jogadores, que serão desafiados a concluírem todo o desafio.

1.4 ORGANIZAÇÃO DO TEXTO

Este trabalho está organizado em capítulos. O capítulo 2 apresenta uma revisão de literatura sobre o uso de recursos visuais aplicados à programação, estudo sobre ferramentas e linguagens de programação visual com blocos, e o uso de programação visual no ensino-aprendizagem de programação paralela. O capítulo 3 apresenta a metodologia utilizada para o desenvolvimento deste trabalho. O capítulo 4 apresenta o projeto da plataforma desenvolvida, com explicações sobre os conceitos de programação paralela que foram abordados nos jogos. O capítulo 5 apresenta detalhes sobre a ferramenta escolhida, e desafios e implementação de novos recursos. O capítulo 6 apresenta a ferramenta final chamada de BlocklyPar, e o capítulo 7 apresenta os testes realizados com usuários. Por fim, o capítulo 9 apresenta a conclusão, considerações finais e trabalhos futuros.

2 REVISÃO DE LITERATURA

Neste capítulo, apresenta-se o uso de recursos visuais como apoio no ensino de programação, trazendo exemplos de ferramentas utilizadas em diferentes áreas. Apresenta-se a programação visual utilizando linguagens de programação com blocos, onde são encontradas e quais recursos a oferecem. E, por fim, o uso de programação visual no ensino-aprendizagem de programação paralela.

2.1 PROGRAMAÇÃO VISUAL

Ferramentas de programação visual vêm se popularizando em diferentes áreas da computação, voltadas principalmente para a introdução à programação. Normalmente, essas ferramentas oferecem componentes que realizam ações quando combinados logicamente, ou que criam protótipos e modelos, base para o desenvolvimento de aplicações.

Programação não é uma habilidade mecânica, mas sim uma forma de pensar, de criar soluções para um problema (VICTOR, 2012). Recursos visuais podem ser aliados nesse contexto, pois os programadores tendem a entender melhor o programa quando visualizam sua execução. Um exemplo é que, quando iniciantes em programação querem encontrar falhas no código, imprimem na tela mensagens constantes para ver onde a falha está ocorrendo, abordagem facilitada posteriormente por ferramentas de depuração.

Utilizando programação visual, os programadores podem focar em seus algoritmos, no sentido de seus códigos, sendo mais criativos e otimizando o seu tempo de trabalho. Ações e estruturas que iriam exigir linhas de código são tratadas como um único elemento, usando figuras e diagramas para ensinar conceitos de programação para iniciantes, como funções, classes e encapsulamento (MASCARELL, 2011).

Pode-se citar diversas ferramentas que utilizam esta abordagem de programação, mesmo que pareçam, à primeira vista, bastante diferentes entre si. Por exemplo, o Simu-Link[1] é uma ferramenta para programação visual de modelos, simulações e análises dinâmicas sobre sistemas. É baseado na plataforma de programação MATLAB (GILAT, 2004) e oferece bibliotecas de componentes visuais customizáveis para criação dos diagramas, utilizando arraste e encaixe dos componentes sobre a tela de desenvolvimento. Ele permite a criação de protótipos com mínimo esforço, em comparação ao uso de ambientes científicos com linguagens textuais.

Em outro nicho, o Android Studio[2] é uma ferramenta para o desenvolvimento de aplicativos para o sistema operacional Android. Para projetar os aspectos visuais do apli-

[1] https://www.mathworks.com/products/simulink.html
[2] https://developer.android.com/studio

cativo, a plataforma oferece um recurso que permite a edição manual de componentes como botões e telas, abstraindo a programação textual por trás desta tarefa.

Outro exemplo ainda é o ArduBlock[3], um *software* de programação visual para placas de prototipagem como Arduino, desenvolvido em Java e voltado a iniciantes em programação. O usuário realiza ações em suas placas combinando blocos lógicos de programação através de arraste e encaixe em uma tela de criação.

Essas e outras ferramentas possuem a vantagem de representar visualmente conceitos de computação e destacam-se pela sua capacidade de apresentar a ideia principal, escondendo detalhes quanto à sintaxe e à semântica de linguagens textuais (MASCARELL, 2011). Com isso, acredita-se que essas e outras ferramentas[4] possam auxiliar na compreensão de conceitos abstratos.

Apesar de funcionarem bem para o seu propósito, há que se mencionar que esse tipo de ferramenta apresenta também limitações, advindas da abordagem visual de programação. Por exemplo: a falta de escalabilidade, pois não permitem a criação de aplicações complexas; a falta de espaço em tela para criar soluções maiores utilizando componentes visuais; limitações para edições avançadas com os componentes disponíveis; e a abstração que a programação visual oferece, limitando a compreensão sobre o uso de recursos de *hardware*.

2.2 PROGRAMAÇÃO VISUAL COM BLOCOS

Um recurso utilizado em ferramentas de programação visual são linguagens de programação baseadas em blocos. Elas oferecem componentes visuais no formato de blocos, que abstraem detalhes sintáticos de linguagens textuais, mas que podem realizar ações iguais ou semelhantes. A programação funciona através do arraste e encaixe de blocos coloridos, com formas que podem se encaixar ou não, limitando o programador a erros, semelhante à montagem de um quebra-cabeça.

Normalmente, ferramentas de programação visual com blocos oferecem atividades lúdicas no formato de jogos e tutoriais, com uma tela de criação, para o desenvolvimento dos algoritmos, e uma tela de visualização, que mostra a saída do programa, como vistos na Figura 2.1. Além disso, elas também costumam agrupar blocos semelhantes na biblioteca de blocos disponíveis, facilitando a escolha de blocos por categorias, como blocos matemáticos, textuais e lógicos (WEINTROP; WILENSKY, 2015).

Essas ferramentas são utilizadas para introdução ao pensamento computacional e à programação de computadores no ensino básico ou na universidade. Em RIZVI et al. (2011), é relatado o uso do Scratch em uma disciplina anterior à de Introdução à Progra-

[3] http://blog.ardublock.com/

[4] https://github.com/ivanreese/visual-programming-codex

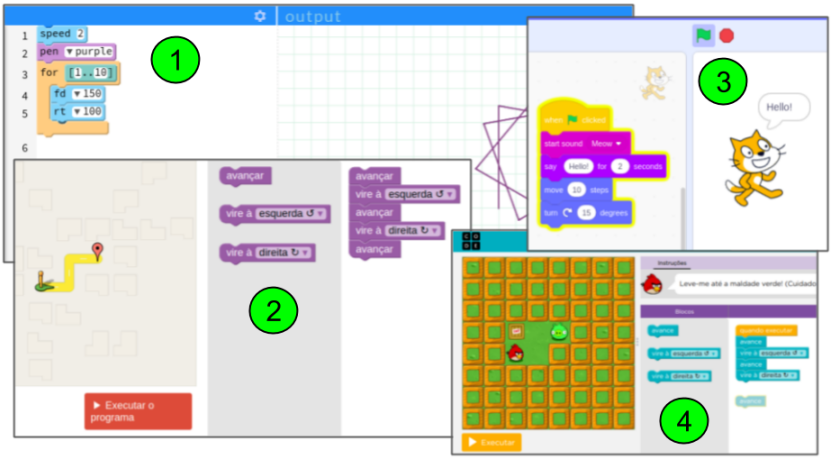


Figura 2.1 – Exemplos de ferramentas para programação visual com blocos: (1) PencilCode, (2) Blockly-Games, (3) Scratch e (4) Code.org.

mação em um curso superior. O uso de Scratch[5] facilitou o aprendizado posterior de programação textual, e mais alunos foram aprovados no curso de Introdução à Programação, 66% comparado à média de 43% nos anos anteriores. Em ROY (2012), relata-se o uso do App Inventor[6] em um curso de verão para alunos de ensino médio, obtendo aumento de interesse desses alunos por cursos de computação.

A Tabela 2.1 apresenta um comparativo entre algumas das principais ferramentas de programação visual com blocos. Foram escolhidas cinco ferramentas que estão em constantes atualizações e que são amplamente utilizadas em cursos para introdução à programação e ao pensamento computacional. As ferramentas são: Blockly-Games, Code.org, App Inventor, Scratch e PencilCode[7].

O Blockly-Games e o Code.org são ferramentas que oferecem jogos no formato de tutorial, com fases de níveis de dificuldade distintos; App Inventor é uma ferramenta para criação livre de aplicativos para dispositivos com sistema operacional Android; e Scratch e PencilCode são ferramentas para criação livre de animações.

As ferramentas da Tabela 2.1 foram testadas nos navegadores Mozilla Firefox[8] e Google Chrome[9]. Nenhuma ferramenta necessitou de instalação de *software adicional*, porém deve-se considerar que os navegadores utilizados já vêm com a extensão do Adobe Flash Player[10], que permite a exibição de conteúdos multimídia. O desenvolvimento de aplicativos com App Inventor ocorre no navegador, porém, para testá-los uma opção é baixar e instalar localmente um emulador de sistema operacional Android.

Todas as ferramentas geram os resultados quando o usuário clica em um comando, geralmente apresentado como um botão escrito “Executar” e nenhuma mostra os resul-

[5] https://scratch.mit.edu
[6] https://appinventor.mit.edu/
[7] http://pencilcode.net
[8] https://www.mozilla.org/en-US/firefox/
[9] https://www.google.com/chrome/
[10] https://get.adobe.com/flashplayer/

	Blockly-Games	Code.org	App Inventor	Scratch	PencilCode
Executa no navegador	✓	✓	✓	✓	✓
Usa software adicional	✗	✗	✓	✗	✗
Executa automaticamente	✗	✗	✗	✗	✗
Formato de jogo/tutorial	✓	✓	✗	✗	✗
Auto-orientada Antes	✓	✓	✗	✓	✗
Auto-orientada Durante	✓	✓	✗	✗	✗
Conversão blocos para texto	✓	✓	✗	✗	✓
Conversão texto para blocos	✗	✗	✗	✗	✓
Tradução para Português-BR	✓	✓	✓	✓	✗

Tabela 2.1 – Comparação entre ferramentas de programação visuais baseadas em linguagens e programação com blocos.

tados em tempo-real, simultâneo à montagem do algoritmo. Mesmo que o presente trabalho não ofereça tal recurso, pontuou-se esse aspecto pois é visto como mais atrativo aos usuários novos em programação, que podem entender seu código e melhorá-lo mais rapidamente[11].

Foram considerados recursos de auto-orientação, apresentados como modais na tela, antes do usuário começar a criar, quando iniciado o ambiente de desenvolvimento, e durante o desenvolvimento. Code.org é a ferramenta que mais oferece modais ao usuário. Blockly-Games oferece em alguns jogos um modal inicial sobre o desafio e em todos oferece orientação durante o desenvolvimento, caso o usuário permaneça muito tempo em uma fase. O Scratch oferece orientação inicial, inclusive com vídeos sobre o seu funcionamento, e o App Inventor e o PencilCode não oferecem modais no ambiente de desenvolvimento.

Apesar de diferenças nas suas funcionalidades e aspectos visuais, por trás da implementação dessas ferramentas são utilizadas as mesmas bibliotecas para a criação de linguagens baseadas em blocos. As bibliotecas mais utilizadas atualmente são Droplet e o Blockly, apresentadas a seguir:

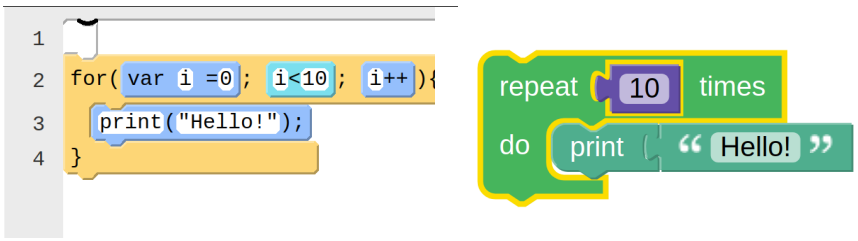
- **Droplet:** Droplet[12] é uma ferramenta de código aberto desenvolvida em 2014, para a criação de linguagens baseadas em blocos. É mantida por organizações como

[11] https://github.com/ezyang/cusec2012-victor/blob/master/transcript.md
[12] https://github.com/droplet-editor/droplet

Code.org e desenvolvedores do PencilCode, e por instituições de ensino como Harvard e MIT. Oferece suporte oficial às linguagens JavaScript, CoffeeScript, HTML e C, e está sendo desenvolvido o suporte à Python, Java e BASIC. O Droplet permite a transição de linguagem visual em linguagem textual, e também o inverso, e a criação de novos blocos de programação. É utilizada no PencilCode e no App Lab[13], um recurso do Code.org. Oferece apenas a versão em inglês, sem tradução para outros idiomas.

- **Blockly:** Blockly[14] é uma biblioteca de código aberto implementada em JavaScript, que facilita a adição de linguagem de programação visual baseada em blocos em aplicativos (PASTERNAK; FENICHEL; MARSHALL, 2017). Ela é atualmente a ferramenta mais popular para esse fim e está por trás de programas como Jogos do Blockly, Hora do Código, MIT App Inventor, Made with Code[15] e, da nova geração do Scratch, uma parceria entre a Google e o MIT Media Labs Scratch Team. O Blockly é utilizado principalmente para a criação de jogos educacionais, permitindo ao desenvolvedor criar sua própria linguagem com blocos. Oferece tradução para diversos idiomas, incluindo Português-brasileiro, e oferece recursos que facilitam a transição de código visual em código textual em JavaScript, Lua, PHP, Dart e Python.

A Figura 2.2 apresenta um mesmo exemplo de laço de repetição que imprime dez vezes a frase “Hello World” com o Droplet e com o Blockly. Nota-se que os blocos do Droplet se assemelham mais à sintaxe usual da linguagem textual original (JavaScript) do que os blocos do Blockly, que utilizam palavras mais próximas da linguagem falada, por exemplo, transformando o laço `for` em um comando “repita”.



((a)) Droplet ((b)) Blockly

Figura 2.2 – Bibliotecas de programação visual com blocos.

[13]https://code.org/educate/applab
[14]https://github.com/google/blockly
[15]https://www.madewithcode.com/

2.3 ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO PARALELA

Em muitos cursos de computação, os alunos aprendem apenas programação sequencial, com enfoque em estruturas de dados e na funcionalidade dos seus algoritmos. Após, se desejarem, podem cursar disciplinas de programação paralela, muitas vezes evitadas por serem consideradas “difíceis”, já que os alunos costumam ter pouca experiência com a exploração de sistemas paralelos, pouco contato com códigos paralelos ou pouca lembrança sobre conteúdos de arquitetura de computadores.

Existem abordagens para o ensino de programação paralela desde o início do ensino de programação. NEZU (2015) propõe uma breve prática sobre programação paralela em ambientes Linux com sistemas *multicore* para ser aplicado em cursos de programação de nível intermediário. A atividade consiste em implementar uma ordenação na linguagem C, serial e paralela, utilizando o padrão OpenMP, e, ao final, orientar os alunos a medirem o tempo de execução da abordagem paralela e da serial, compará-las e explicar os resultados.

Além disso, foram encontrados trabalhos que propõem novas linguagens de programação textuais, criadas para facilitar o ensino de programação paralela a iniciantes em computação (BURKE, 2015; GREGG et al., 2012). EcoSim é uma linguagem próxima da língua inglesa criada para incentivar o pensamento paralelo entre os usuários, iniciantes em programação. Ela foi utilizada em um curso de 5 dias para alunos de 9 e 10 anos sem conhecimento prévio de programação, e foi executada em um ambiente *online* próprio, que gera relatórios sobre cada execução. Já Chapel, é uma linguagem semelhante à Python e Java, que oferece comandos que facilitam a implementação de tarefas em paralelo.

A programação paralela desperta o interesse do programador em dar atenção aos recursos subjacentes relacionados às arquiteturas dos computadores. Porém, como nem sempre este conhecimento é essencial para a execução correta dos programas, muitas vezes é deixado de lado. Com isso, o ensino de programação paralela concomitantemente ao ensino de conceitos básicos de programação pode instigar os desenvolvedores a criar programas mais eficientes desde o princípio.

Em FENG; TILEVICH; FENG (2015), é apresentado um projeto para ensinar programação paralela através da adição explícita de abstrações paralelas em linguagens baseadas em blocos. Para isso, os autores utilizaram o Snap! e bibliotecas que auxiliaram na percepção de paralelismo durante a execução dos programas (Feng; Feng, 2016). Ao todo, foram criados três novos blocos que realizam funções já existentes mas em paralelo, e duas atividades para utilizar esses blocos.

O público-alvo desse projeto foi composto por alunos de ensino básico e pesquisadores, de qualquer nível de ensino, não necessariamente da computação. Nesse contexto, os autores criaram uma solução para converter programas criados no Snap! para código textual, pronto para ser executado em ambientes de programação tradicionais, ex-

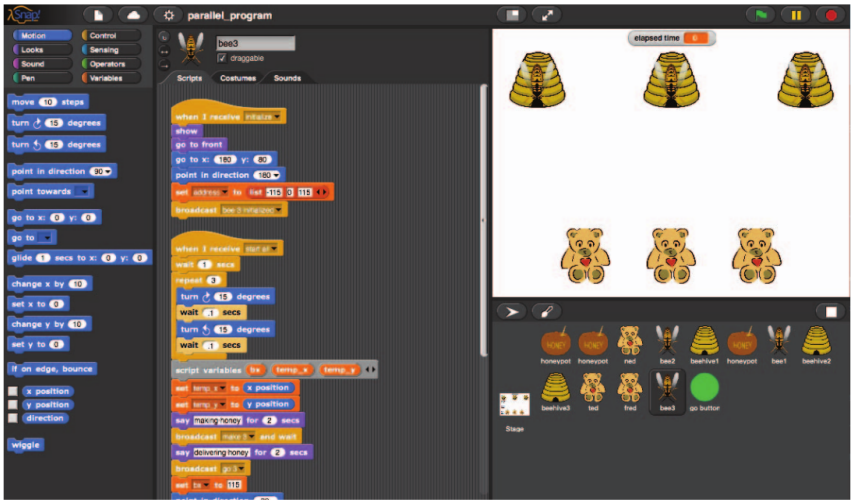


Figura 2.3 – Imagem retirada de (FENG; TILEVICH; FENG, 2015), que apresenta uma atividade criada no Snap! para abordar o paradigma do produtor-consumidor.

pressando o paralelismo com a adição de pragmas OpenMP (FENG; GARDNER; FENG, 2017).

A primeira atividade utilizando os blocos foi a resolução do problema do produtor-consumidor, um desafio clássico de computação que requer sincronização de processos. Ele foi apresentado de forma lúdica, onde os produtores são abelhas, o produto é o mel, e os consumidores são ursos (Figura 2.3). Outra abordagem apresenta *sprites* de uma jarra e três copos, onde o usuário pode servir um copo por vez ou os três copos ao mesmo tempo, buscando desenvolver a noção de como o paralelismo pode acelerar operações em comparação à execução serial.

Os autores avaliaram a eficácia dos novos blocos em uma atividade prática com alunas de ensino médio. As alunas realizaram construções livres para as suas soluções, inicialmente utilizando blocos sequenciais, para depois reconstruí-las com blocos paralelos.

Os autores desse trabalho inspiraram-se no Scratch, uma ferramenta para a criação livre de animações e jogos utilizando blocos de programação. Porém, supõe-se que utilizar tutoriais em formato de jogo, com fases pré definidas, guiadas através de mensagens com modais, como apresentados na Hora do Código e nos Jogos do Blockly, tornariam a atividade de programação com blocos mais atrativa e desafiadora ao usuário, além de independente de orientação para sua resolução.

3 METODOLOGIA

Este trabalho foi guiado pelo método conhecido como *Design Science Research (DSR)* (PIMENTEL; FILIPPO; SANTORO,). Desenvolveu-se um artefato para resolver um problema observado, inserido em um contexto específico, gerando conhecimento que foi avaliado para identificar melhorias e refinar o projeto.

A programação visual com blocos tem sido apresentada como uma abordagem eficaz para introdução à programação, pois abstrai do programador preocupações com a sintaxe de linguagens textuais. Partindo da premissa de que o ensino de programação paralela pode ser facilitado pelo uso desse recurso, criou-se um artefato chamado BlocklyPar, que oferece 3 jogos para praticar o pensamento paralelo.

Os jogos do BlocklyPar têm o formato de tutorial, com fases predefinidas nas quais o usuário é guiado a resolver desafios que abordem paralelismo. Os desafios buscam apresentar a ferramenta de programação com blocos, conceitos de programação paralela e incentivar a aplicação de paralelismo de forma gradual, frisando a comparação entre o uso de abordagens seriais e o uso de abordagens paralelas.

Após definir o objetivo final deste trabalho, planejou-se o seu desenvolvimento, levantando os requisitos necessários e dividindo esse processo em estágios. Para isso, utilizou-se uma metodologia ágil, criando um *software* adaptável e passível a mudanças e melhorias a qualquer momento, conforme observado nos testes realizados durante a implementação e após testes com usuários. A implementação seguiu um método inspirado na Extreme Programming (XP), com ciclos curtos de desenvolvimento e objetivos revisados semanalmente.

Os jogos foram desenvolvidos utilizado git[1] para versionamento da implementação, com repositório público disponível em: https://github.com/blocklypar. O BlocklyPar foi hospedado em um site com informações sobre a plataforma e sobre os jogos, disponível em: https://blocklypar.github.io.

No primeiro ciclo de desenvolvimento, após realizar estudos sobre as ferramentas disponíveis e sobre os conceitos de programação paralela que se pretendia abordar, foram criados protótipos das fases de cada jogo. No segundo ciclo, iniciou-se a implementação dos jogos, baseada no código-fonte do jogo Maze[2], do Blockly-Games. Cada fase implementada foi observada no contexto do jogo e da plataforma, para verificar se ela fazia sentido e se conseguia expressar os conceitos de paralelismo que se propôs.

Em consonância com o método DSR, o segundo ciclo foi permeado pela realização de testes com usuários estudantes de Ciência da Computação e Sistemas de Informação da Universidade Federal de Santa Maria. Os usuários foram divididos em dois grupos: 2

[1] https://git-scm.com/

[2] https://blockly.games/maze

alunos de pós-graduação especialistas em programação paralela e 11 alunos de graduação sem conhecimento aprofundado de programação paralela.

Os usuários jogaram todos os jogos disponíveis no BlocklyPar, orientados a acessarem a tela com explicações sobre a plataforma antes de iniciarem as jogadas, e de que anotassem críticas e sugestões pois seria aplicado um formulário avaliativo ao final. O formulário teve questões de múltipla-escolha e questões dissertativas, com 14 questões para o primeiro grupo e 12 para o segundo.

Primeiro, foram realizados os testes com os alunos de pós-graduação, que jogaram e responderam o formulário à distância. Com os resultados obtidos, foram implementadas melhorias e correções nos jogos para o segundo teste, que ocorreu presencialmente com 8 dos 11 alunos de graduação.

O formulário para os especialistas em programação paralela foi mais voltado à usabilidade do jogo, com questões sobre os conceitos de paralelismo abordados pelo BlocklyPar. Para os alunos de graduação, as questões foram voltadas à abstração de paralelismo obtida pelos recursos dos jogos, e a aceitação do público-alvo. Estas respostas foram essenciais para verificar a eficiência do BlocklyPar e coletar sugestões para trabalhos futuros.

4 PROJETO DA PLATAFORMA

Considerando o público-alvo como estudantes de cursos superiores de Computação, a proposta da plataforma é abordar conceitos e técnicas utilizadas em programação paralela aplicadas em tarefas cotidianas de estudantes universitários, como ir para a sala de aula e entregar livros na biblioteca.

Para isso, foram criados três jogos, cada um com fases, objetivo e contexto distintos. Os jogos apresentam gradualmente o ambiente de programação com blocos, os conceitos de paralelismo e a utilização de recursos para implementar programação paralela.

Muitos dos estudantes de Computação não têm contato com ferramentas de programação com blocos, e por isso não estão acostumados com a dinâmica de arraste e encaixe. Visto isso, o primeiro jogo introduziu a linguagem de programação com blocos e o fluxo de execução do jogo de maneira sequencial.

A partir do segundo jogo, foram introduzidos conceitos de programação paralela, abordados em áreas de estudos como Computação de Alto Desempenho. Esses conceitos são apresentados à seguir:

4.1 REALIZAÇÃO DE TAREFAS

Durante cursos superiores de computação, os alunos aprendem a utilizar estratégias algorítmicas para resolver seus problemas através de programação, como implementar programas para buscar e ordenar dados. Apesar dessas aplicações terem objetivos distintos, todas precisam realizar tarefas sobre dados de entrada e gerar um resultado.

Em computação, as tarefas são atribuídas aos recursos disponíveis para executá-las. Esses recursos podem ser diferentes processadores, diferentes núcleos de processadores *multicore*, ou diferentes *threads* de um processo. Para situações com apenas um recurso disponível, as tarefas executam de forma concorrente, onde o recurso lida com cada tarefa, uma de cada vez. Em situações com mais recursos disponíveis, as tarefas podem ser executadas em paralelo.

Cada tarefa é responsável pela execução sobre um conjunto de dados, podendo ser o conjunto de dados total ou parte dele. Caso exista mais de uma tarefa ela pode ter sua execução dependente ou não de outra. Tarefas dependentes são executadas de forma serial, enquanto tarefas sem dependência tem potencial para serem executadas ao mesmo tempo.

4.2 PARALELISMO

Atualmente, existe alta demanda por maior poder computacional para processar grandes quantidades de dados em tempos de execução razoáveis. Ambientes paralelos podem ser, por exemplo, processadores *multicore* ou vários computadores interconectados, todos com recursos disponíveis para a execução de programas.

O paralelismo é uma abordagem utilizada para otimizar o desempenho de programas, permitindo com que diferentes tarefas sejam processadas ao mesmo tempo. Para a realização de tarefas em paralelo, o trabalho total é dividido e atribuído a cada tarefa, que vai realizar o seu trabalho ao mesmo tempo que as outras. Cada tarefa gera um resultado sobre a porção de dados trabalhada, que faz parte da solução total final. Essa abordagem contribui para a execução mais rápida do trabalho total, se comparado à execução de todo o trabalho por uma única tarefa, sequencialmente.

Existem diferentes técnicas para a criação de programas paralelos. Em 1995, Ian Foster propôs uma metodologia que consiste em (i) particionar os dados, entrada da aplicação, em pedaços independentes, (ii) determinar como as tarefas irão se comunicar, (iii) reagrupar tarefas que realizam pouco trabalho em tarefas maiores, e (iv) mapear as tarefas finais para serem executadas em diferentes nodos do processador (FOSTER, 1995).

Essa técnica mostra a importância de se analisar a aplicação antes de inserir abordagens paralelas. Com isso, buscou-se inserir gradualmente nos jogos os elementos necessários para a criação de uma solução paralela: o contexto da aplicação, a tarefa a ser realizada, o objetivo com o resultado esperado, os dados, e o recurso utilizado para as execuções.

4.3 MÉTRICA DE EXECUÇÃO

Métricas de execução são unidades para mensurar o comportamento de uma situação. Em programação paralela, uma métrica utilizada em aplicações é a obtenção do tempo de relógio gasto na execução da versão paralela em comparação à versão serial, ao computarem os mesmos dados. Outra medida utilizada é o *timestep*, unidade atribuída a cada ação de um programa, que ao final somam o tempo total gasto para a execução de todas as tarefas.

O *speedup* é uma medida de desempenho que utiliza os resultados obtidos pelas métricas de tempo de execução, mostrando o quão mais rápida foi a execução utilizando multiprocessador em comparação com a melhor execução sequencial. O caso ideal ocorre quando o uso de n processadores é n vezes mais rápido do que a utilização de um único processador, o que não costuma acontecer em aplicações reais que gastam tempo com outros processamentos, como transferências de dados e sincronização de processos.

Essas medidas colaboram para uma melhor visualização sobre a execução do programa, podendo identificar pontos de melhorias no uso do paralelismo em comparação à implementação sequencial. Além disso, as métricas podem ser utilizadas posteriormente para análises mais detalhadas sobre o desempenho do programa.

4.4 ANÁLISE DE DESEMPENHO

Após a coleta das métricas de execução do programa paralelo, podem ser realizadas análises de desempenho sobre o programa. Com as análises pode-se identificar se a abordagem obteve um tempo de execução menor quando utilizou mais recursos, se ela explorou bem os recursos disponíveis e se ela poderia ser implementada de forma menos custosa, por exemplo.

Em programação paralela, esta é uma etapa contínua, onde o desenvolvedor realiza vários experimentos sobre o seu programa, com execuções utilizando diferentes números de recursos e diferentes parâmetros, obtendo informações variadas sobre suas execuções. Após, os dados obtidos são manipulados, realizando cálculos para medir o desempenho do programa, como o cálculo do *speedup*, e, por fim, modificar a implementação para obter melhor desempenho.

5 IMPLEMENTAÇÃO

Este capítulo apresenta a implementação dos jogos. É apresentada a linguagem de programação com blocos escolhida, a ferramenta que serviu como inspiração para o desenvolvimento deste trabalho, os novos recursos implementados para expressar o paralelismo nos jogos, e para coleta de informações sobre o acesso à plataforma.

5.1 BLOCKLY

Blockly é uma biblioteca de código aberto lançada em 2012 pela Google para adicionar linguagem baseada em blocos em aplicativos (Pasternak; Fenichel; Marshall, 2017). Ela é escrita em JavaScript, permitindo a tradução direta de linguagem com blocos para linguagem textual em JavaScript, e oferecendo um ambiente que permite explorar a tradução para outras linguagens [1].

O principal recurso oferecido pela biblioteca é a possibilidade de criação e edição de novos blocos em diferentes linguagens. Para criar um bloco de uma nova categoria deve-se definir um "gerador de código" ao bloco que se deseja criar. A ferramenta já oferece os códigos para blocos textuais, de repetição, variáveis (estáticas e dinâmicas), de procedimento, matemáticos, lógicos, para edição de cores, e para definição de listas.

Para criar novas categorias, o desenvolvedor deve definir o bloco criado, configurando aspectos visuais do mesmo, e associando o código em linguagem textual que será atribuído ao uso daquele bloco. Conforme visto na Figura 5.1, à esquerda estão as funções para definir uma variável `set x to` e para obter o seu valor `x`, e à direita está o resultado dessa implementação nos blocos criados.

```
// Variable setter.
Blockly.JavaScript['variables_set'] = function(block) {
  var argument0 = Blockly.JavaScript.valueToCode(block, 'VALUE',
      Blockly.JavaScript.ORDER_ASSIGNMENT) || '0';
  var varName = Blockly.JavaScript.variableDB_.getName(
      block.getFieldValue('VAR'), Blockly.Variables.NAME_TYPE);
  return varName + ' = ' + argument0 + ';\n';
};

// Variable getter.
Blockly.JavaScript['variables_get'] = function(block) {
  var code = Blockly.JavaScript.variableDB_.getName(block.getFieldValue('VAR'),
      Blockly.Variables.NAME_TYPE);
  return [code, Blockly.JavaScript.ORDER_ATOMIC];
};
```

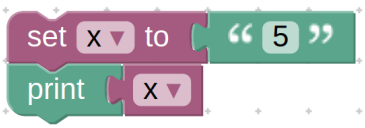


Figura 5.1 – Funções em JavaScript para criar novos blocos e os blocos criados.

Além do Blockly, o Google oferece o Blockly-Games, uma coleção de 8 jogos para introdução à programação utilizando linguagens com blocos. O Blockly-Games também

[1] https://github.com/google/blockly

tem código aberto disponível em um repositório hospedado no GitHub[2]. Cada jogo segue o formato de tutorial, com fases de diferentes níveis de dificuldade.

Para o desenvolvimento deste trabalho, utilizou-se como base o jogo Maze do Blockly-Games. O objetivo deste jogo é orientar um personagem a percorrer um labirinto até chegar no seu destino utilizando uma coleção de diferentes blocos disponíveis em cada fase. Um ponto forte desta e de outras fases é que a introdução de conceitos mais avançados, como condicionais e blocos de repetição, é feita gradualmente, facilitando o aprendizado do usuário.

Outro ponto é o uso de *pop-ups* de orientação (Figura 5.2) durante o primeiro contato com cada fase, que são lançados mediante alguma condição observada pela interação do usuário no jogo, como tempo de inatividade ou uso de blocos em locais incorretos da tela. Além disso, o jogo também utiliza modais ao final de cada fase (Figura 5.3) parabenizando o usuário e apresentando a representação do código submetido em linguagem textual.

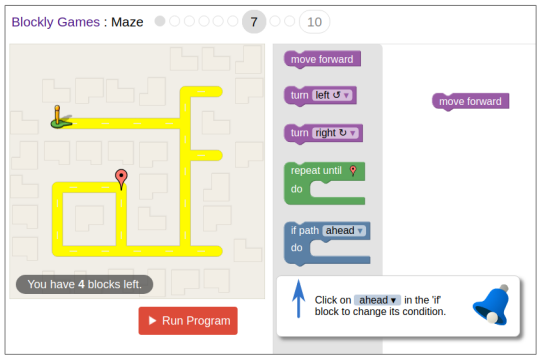


Figura 5.2 – Pop-up no Blockly-Games.

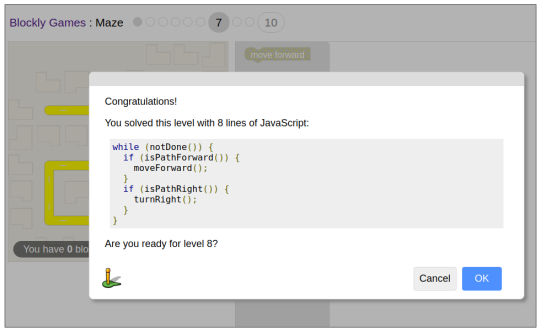


Figura 5.3 – Modal no Blockly-Games.

No repositório do Blockly-Games, há instruções sobre a compilação do projeto, utilizando um arquivo de compilação `Makefile` com as opções de compilar todos os jogos para o inglês, compilar um jogo específico para o inglês ou compilar um jogo para todas as mais de 50 linguagens disponíveis. Durante a compilação são requeridos alguns *software* adicionais: svn (cliente para subversão), Java, sed, unzip e Python. Notou-se que o Python deve ser de uma versão inferior à versão 3, o que não é documentado no projeto original.

A ferramenta utiliza templates Closure[3], também conhecido como Soy Templates, um sistema de templates desenvolvido pela Google que executa do lado do cliente e do servidor para gerar dinamicamente código HTML e elementos da interface do usuário em linguagem reconhecida para Java e JavaScript. Diferente de sistemas de template tradicionais, onde deve-se criar um arquivo por página web, o Closure Templates preza a fácil reutilização dos códigos, permitindo a criação de partes do template em diferentes arquivos, que ao compilados produzem a interface final do usuário. Atualmente, eles são

[2]https://github.com/google/blockly-games/
[3]https://github.com/google/closure-templates

utilizados em aplicações web da Google como Gmail e Google Docs, e funcionam em qualquer ambiente de aplicativos Web independente das bibliotecas, estruturas ou outras ferramentas.

Um benefício dos templates Closure é que eles são pré-compilados em JavaScript, o que aumenta o desempenho da aplicação no lado do cliente. Além disso, a sintaxe utilizada pelos templates visa facilitar a edição, mas preservar a visualização da estrutura do HTML gerado, facilitando a leitura de quem já está acostumado com a linguagem. As mensagens da página web que precisam passar por tradução, por exemplo, são definidas individualmente e podem ser acessíveis em outros arquivos, para fácil leitura do usuário.

5.2 CONFIGURAÇÕES INICIAIS

Visando agilizar o desenvolvimento dos jogos, modificou-se o arquivo `Makefile` para ele compilar apenas os jogos necessário e o "index" (menu inicial) em inglês. Visto que originalmente os principais aspectos sobre o funcionamento do jogo Maze foram implementado em um único arquivo JavaScript, a segunda ação foi realizar a modularização dos arquivos.

Este trabalho foi desenvolvimento em um sistema UNIX, com a distribuição Ubuntu 18.04.3 LTS, e utilizando o navegador Google Chrome para testes. Além do código-fonte, utilizou-se o fórum oficial do Blockly para possíveis dúvidas na implementação do trabalho[4].

O Blockly oferece uma biblioteca de blocos prontos para utilização, mas todos são utilizados na programação sequencial de programas. Para concretizar as fases projetadas e expressar o paralelismo de forma clara ao usuário, seriam necessários blocos com aspecto visuais e funcionais diferentes dos existentes. Assim, foram criados novos blocos de programação.

5.3 CRIAÇÃO DE NOVOS BLOCOS

O primeiro bloco criado foi o de realização de tarefas. A ideia desse bloco é executar repetidamente todas ações dos blocos encaixados nele, porém, verificando a cada iteração se a tarefa determinada foi satisfeita. Por exemplo, se o segundo bloco da Figura 5.4 realizará as ações relacionadas à ele até que todos os livros tenham sido entregues, conforme descrito no bloco, quando `livros == 0`.

O próximo bloco criado foi o de execução de tarefas em paralelo, apresentado na Figura 5.5. Assim como o bloco de repetição, e o bloco de realização de tarefas sequenci-

[4]https://groups.google.com/forum/!forum/blockly

Figura 5.4 – Blocos para a realização de tarefas.

almente, ele irá executar todas as tarefas inseridas nele, porém, agora ele poderá executar a mesma tarefa sobre diferentes recursos ao mesmo tempo, em paralelo.

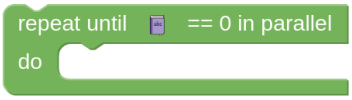


Figura 5.5 – Bloco para realização de tarefas em paralelo.

Para determinar quais recursos estão disponíveis para a execução de tarefas em paralelo, foi criado um novo bloco, apresentado na Figura 5.6. Este bloco, define os estudantes que irão executar as ações, comparando com programação paralela: o recurso disponível que será alocado para executar determinada tarefa.

Este bloco traz a noção de atribuição, onde o usuário deve definir o estudante que quer movimentar, utilizando o *dropdown* do bloco. Para movimentar o estudante, deve-se encaixar os blocos lógicos dentro deste novo bloco, e caso se utilize mais de um bloco de estudante, deve-se encaixá-los no bloco de realização de tarefas em paralelo para que sejam executados ao mesmo tempo.

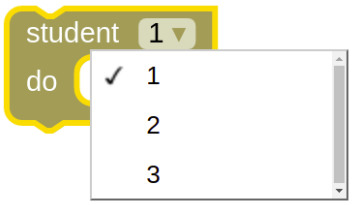


Figura 5.6 – Bloco para atribuição do estudante que realizará as ações.

5.4 MÉTRICA DE DESEMPENHO

Para expressar o tempo de execução gasto nas fases que abordam paralelismo, é utilizado um contador (Figura 5.7) que fica abaixo da tela de animação, em vermelho, e é zerado no início da fase ou ao reiniciá-la. Essa métrica de desempenho é utilizada para expressar a diferença de uso do paralelismo em comparação à execução serial.

Ao invés de se utilizar o tempo do relógio, associou-se a cada bloco um tempo de execução. A utilização de um bloco “avance”, por exemplo, contabiliza 1 unidade de tempo,

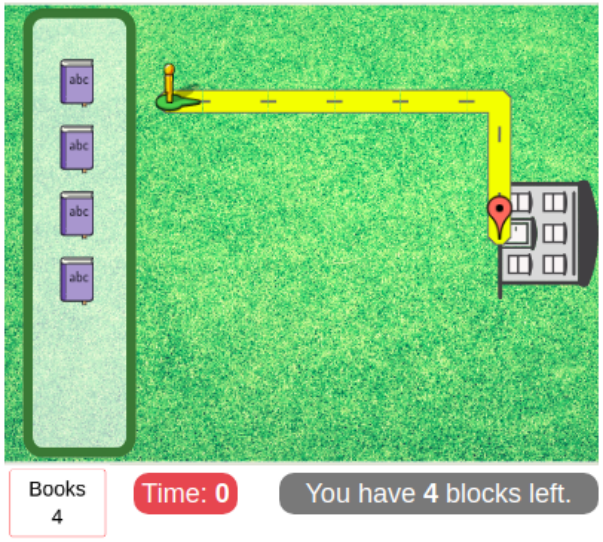


Figura 5.7 – A métrica de desempenho é expressa pelo contador em vermelho na tela.

enquanto a ação de verificar se existe um caminho, chamada no bloco condicional “if - else”, contabiliza 0.5 unidades de tempo. Com isso, o tempo de execução de uma fase fica conciso para o usuário poder comparar abordagens seriais ou paralelas usando mais ou menos recursos.

5.5 PROCESSOS EM PARALELO

Programas na linguagem de programação JavaScript executam do lado do cliente, em um navegador web. No navegador, cada aba executa o seu código em uma *thread*. Os navegadores atuais executam código JavaScript de forma tão rápida que o cliente não percebe quando uma solicitação web bloqueia toda a página. Entretanto, para solicitações mais custosas esse bloqueio pode ser percebido e indesejado.

A alternativa mais usada para inserir paralelismo em execuções com JavaScript é o uso de *Web Workers*, uma API oferecida pela maioria dos navegadores, que particiona a *thread* em outras, chamadas de *workers*. Apesar de eficaz em trazer paralelismo, essa técnica não traz ganho de desempenho, apenas permite que o site responda a nível de usuário de forma paralela.

O Blockly utiliza um interpretador[5] criado por um de seus desenvolvedores como alternativa a utilização de *Web Workers*. Cada interpretador criado lê uma trilha de blocos encaixados e executa as ações dos blocos de cada trilha paralelamente.

Foi estudada a possibilidade de usar mais de um interpretador para executar mais de uma trilha de blocos em paralelo, porém, preferiu-se manter a execução concorrente a nível de servidor, com a ideia de paralelismo a nível de usuário. Acreditou-se que além de

[5]https://neil.fraser.name/software/JS-Interpreter/docs.html

maior tempo para implementação, pois não foram encontrados exemplos onde múltiplos interpretadores foram aplicado no contexto do Blockly, ele poderia gerar confusão com iniciantes e não iniciantes na plataforma, pois qualquer bloco solto na tela de criação ficaria passível a ser considerado como uma trilha de execução.

Assim, os processos relacionados aos estudantes executam em paralelo do lado do usuário, mas de forma concorrente do lado do servidor. Essa representação com os blocos foi feita ao atribuir tarefas a diferentes estudantes dentro do bloco `repeat until ... in parallel`, declarando um estudante abaixo do outro. Essa representação, com apenas uma trilha de execução, se assemelha à declaração de sessões com OpenMP.

Utilizando o `#pragma omp sections` do OpenMP são chamadas as *threads* disponíveis no sistema, e as tarefas de cada seção, definidas pelo `#pragma omp section`, são atribuídas a uma *thread*, que executa em paralelo. Diferente do uso de seções que executam apenas uma vez cada bloco da seção, o bloco executa continuamente até finalizar a tarefa. Esta relação entre linguagem com blocos e linguagem textual pode ser vista na Figura 5.8, onde à esquerda foi implementado em C o escopo das seções com OpenMP, e à direita os blocos utilizados no jogo.

```
#pragma omp parallel{
    #pragma omp sections{
        #pragma omp section{
            // definição da seção 1
        }
        #pragma omp section{
            // definição da seção 2
        }
        #pragma omp section{
            // definição da seção 3
        }
    }
}
```

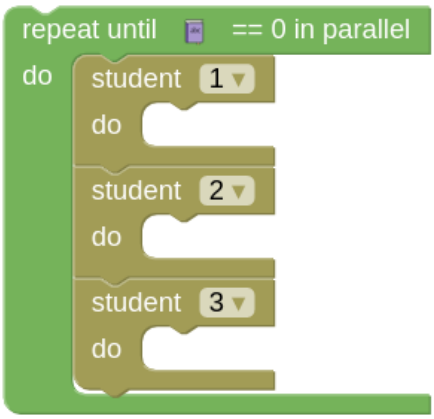


Figura 5.8 – Comparação entre o uso de paralelismo com OpenMP e o uso de paralelismo no BlocklyPar.

5.6 OUTROS RECURSOS UTILIZADOS

A seguir, são apresentados outros recursos utilizados na implementação da plataforma, que contribuíram para o funcionamento dos jogos e demais melhorias.

5.6.1 Aspecto dos jogos

Os jogos propostos na nova plataforma possuem aspectos visuais semelhantes ao do jogo Maze do Blockly-Games, visto na Figura 5.9. À direita fica a tela de criação, onde o usuário combina os blocos para gerar a resposta do desafio, e à esquerda fica a tela de execução, que é animada de acordo com o programa construído com os blocos. Para isso, o usuário deve clicar no botão “Run Program”.

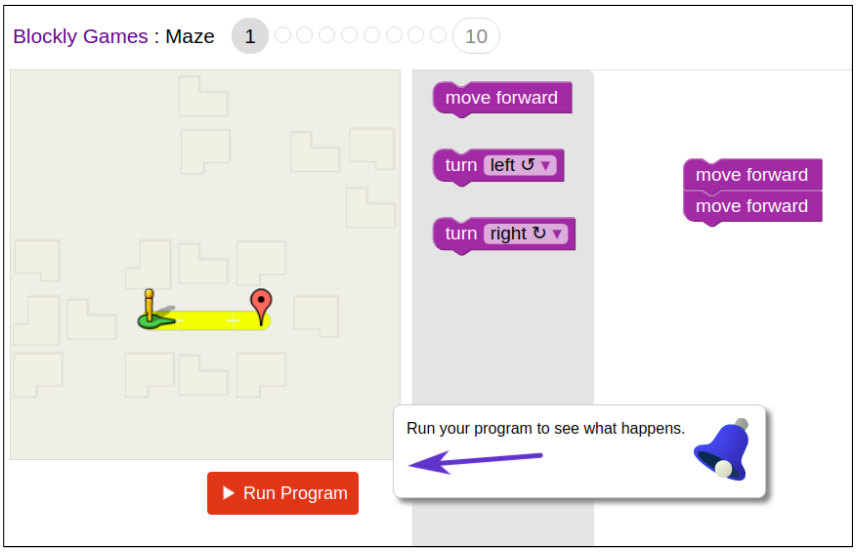


Figura 5.9 – Fase 1 do jogo Maze do Blockly-Games.

Além disso, os novos jogos mantiveram a ideia de caminho a ser percorrido, como um labirinto, visto que esse contexto é propício a utilização de blocos lógicos variados. Porém, ao invés do usuário orientar um personagem até o objetivo final e vencer o jogo, o usuário deve orientar o personagem a realizar todas as tarefas para concluir a fase.

5.6.2 Armazenamento de dados

Buscando coletar informações sobre o acesso à plataforma, foi utilizado o Google Analytics[6]. Para utilizá-lo, deve-se criar um perfil para a aplicação a ser analisada e inserir uma *tag* no cabeçalho dos arquivos HTML que se deseja monitorar.

A página inicial sobre a aplicação associada ao Google Analytics fica disponível conforme visto na Figura 5.10. Nela, pode-se obter informações sobre o número de usuários que acessaram as páginas, quais páginas foram acessadas, o tempo de permanência nas páginas, a localização geográfica dos usuários, o horário de acesso, o link original que redirecionou o acesso à outras páginas, etc.

Acredita-se que as tentativas de resposta submetidas pelo usuário são tão importantes quanto as respostas corretas submetidas para análises sobre a jogabilidade da

[6]https://analytics.google.com/

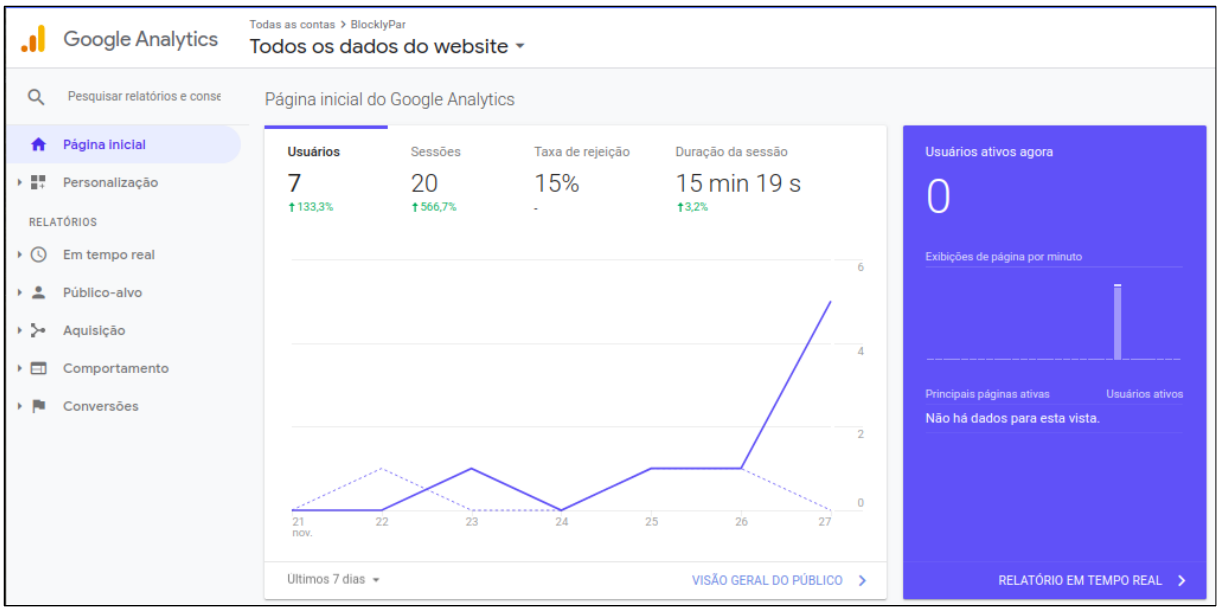


Figura 5.10 – Página inicial do Google Analytics para monitorar o BlocklyPar.

plataforma. Porém, o Google Analytics não oferece informações detalhadas sobre o comportamento do usuário nas páginas acessadas.

Caso fosse executado localmente, os navegadores atuais costumam armazenar em cache ações do usuário em sites, e com isso, poderia se obter maiores informações sobre a interação do usuário com as páginas. Atualmente, o BlocklyPar, assim como o Blockly-Games, está configurado para permitir o apenas o armazenamento das respostas corretas submetidas pelo usuário, e não o das tentativas falhas.

5.6.3 Tradução

A tradução dos textos dos blocos do Blockly e do Blockly-Games para outros idiomas é realizado com o Translatewiki[7]. Para isso, deve-se inserir os textos no idioma padrão utilizando *tags* que identifiquem o bloco que se quer descrever. Criando o perfil da plataforma que se quer traduzir no Translatewiki os usuários podem colaborar com a tradução aberta e gratuita para outros idiomas.

Para traduzir um jogo, deve-se compilá-lo individualmente para cada idioma desejado. Visto isso, o projeto apresentado neste trabalho foi implementado e está disponível apenas em inglês, linguagem original do Blockly e do Blockly-Games. Com isso, as etapas de implementação e compilação puderam ser realizadas sem atrasos.

[7] https://translatewiki.net/wiki

6 BLOCKLYPAR

A plataforma final recebeu o nome de BlocklyPar, visto que utiliza a biblioteca Blockly para sua implementação e tem o objetivo de introduzir programação paralela. A sua proposta é incentivar a prática do pensamento paralelo e ensinar conceitos de paralelismo através de jogos que envolvam tarefas do cotidiano de estudantes universitário, público-alvo da plataforma.

O BlocklyPar é dividido em três jogos com desafios distintos. O usuário deve programar com blocos as ações de um personagem para percorrer um caminho até completar o desafio proposto. Cada jogo possui quatro fases com níveis de dificuldade que aumentam gradualmente, e cada fase possui uma biblioteca de blocos à disposição para criação do programa, sendo esses os blocos de programação usuais do Blockly, e os novos blocos implementados.

As fases de cada jogo foram definidas buscando instigar o uso de paralelismo em programação de computadores, voltado a alunos de ensino superior de cursos de Computação. Os jogos foram criados para que atingissem os seguintes objetivos:

1. Apresentar um ambiente de programação com blocos;
2. Apresentar conceitos usualmente expressos com programação com blocos, como uso de blocos lógicos, blocos de repetição e blocos condicionais;
3. Apresentar o conceito de tarefas através de um novo bloco, que é base para a implementação de aplicações paralelas em programas com linguagem textual;
4. Apresentar o conceito de paralelismo através de novos blocos, mostrando que a tarefa pode ser dividida para ser executada em mais de um recurso ao mesmo tempo.

Visando facilitar a compreensão do usuário, foram inseridos modais no início e no final das fases, e *pop-ups* para apresentação de blocos novos. Os modais de início aparecem em quase todas as fases, apenas quando o usuário ainda não tiver as resolvido, e os modais de final aparecem em todas as fases, caso o usuário ainda não as tiver concluído. Os modais de final apresentam uma mensagem parabenizando o usuário e o orientando à próxima fase, ou jogo.

A Figura 6.1 apresenta a página inicial do BlocklyPar, que contém uma breve explicação sobre os jogos, um link “About” para uma tela com mais informações sobre o projeto, e links para os jogos disponíveis. Cada resposta submetida corretamente fica armazenada localmente na memória da plataforma, assim, caso o usuário já tenha iniciado a resolução de um ou mais jogos e quiser recomeçar, é disponibilizado um botão na tela inicial para “limpar os dados” com as respostas armazenadas.

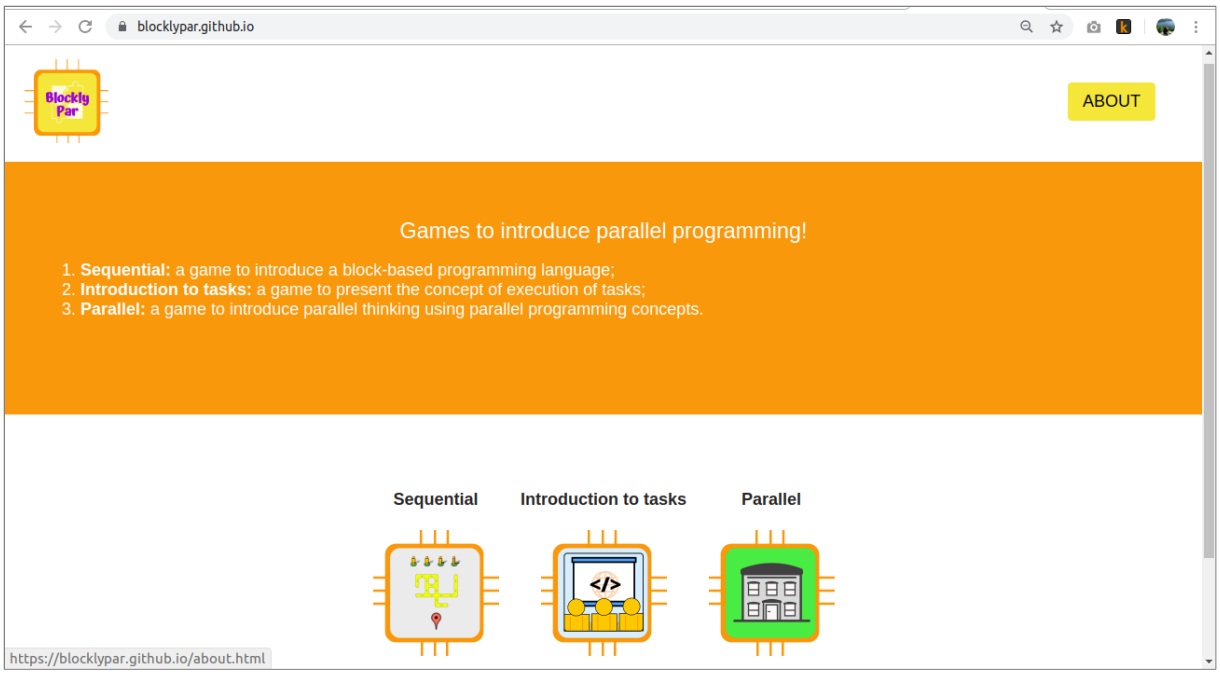


Figura 6.1 – Página inicial do BlocklyPar.

O projeto foi hospedado na plataforma GitHub Pages[1], um serviço que permite a hospedagem gratuita de páginas estáticas, associando um repositório à uma página web de mesmo nome. O código fonte do BlocklyPar está disponível para acesso em: https://github.com/blocklypar. A seguir, são apresentados os três jogos disponíveis, com *screenshots* e explicações sobre o seu funcionamento.

6.1 JOGO 1: PROGRAMAÇÃO SEQUENCIAL

O primeiro jogo busca introduzir a programação visual com blocos utilizando os blocos lógicos de programação básicos da biblioteca Blockly. O objetivo em cada fase é o mesmo do jogo Maze, onde o personagem deve percorrer um labirinto até chegar no seu destino final.

Foram implementadas 4 fases com caminhos distintos e blocos lógicos que são acrescidos gradualmente conforme o decorrer das fases. A Figura 6.2 apresenta a primeira fase, a única deste jogo que contém um modal para explicar como funciona a programação através da combinação de blocos com arraste e encaixe para compor os programas.

A Figura 6.3 apresenta a terceira fase, que dispõe dos novos blocos criados, introduzidos por um *pop-up*. Visto que este jogo é usado apenas como introdução à ferramenta, sem usar conceitos de programação paralela, foram implementadas apenas as quatro fases, que avançaram mais rapidamente no grau de dificuldade em comparação ao jogo

[1] https://pages.github.com/

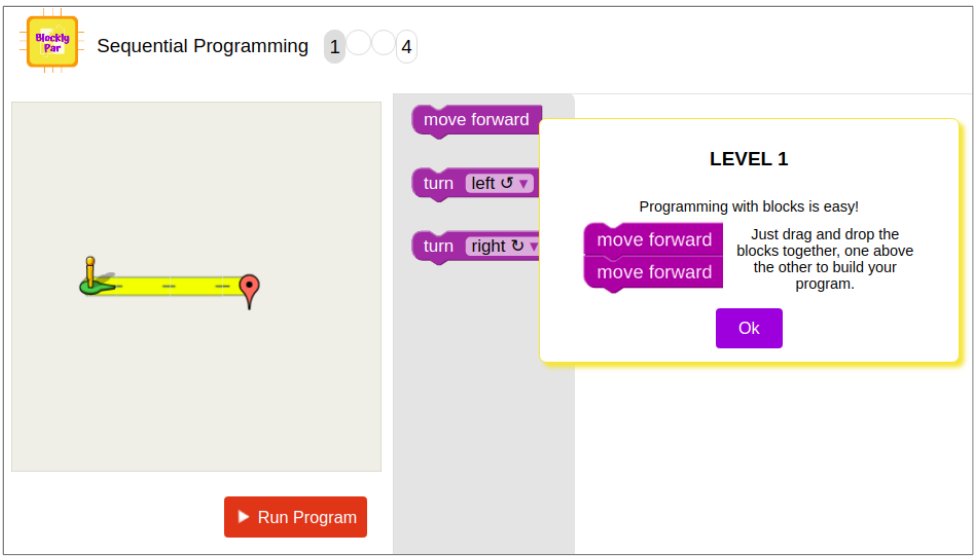


Figura 6.2 – Fase 1 do jogo para introdução à programação com blocos.

Maze do Blockly-Games, que possui 10 fases. Além disso, acelerou-se a execução das ações do personagem no labirinto, visando evitar cansar o usuário para os próximos jogos.

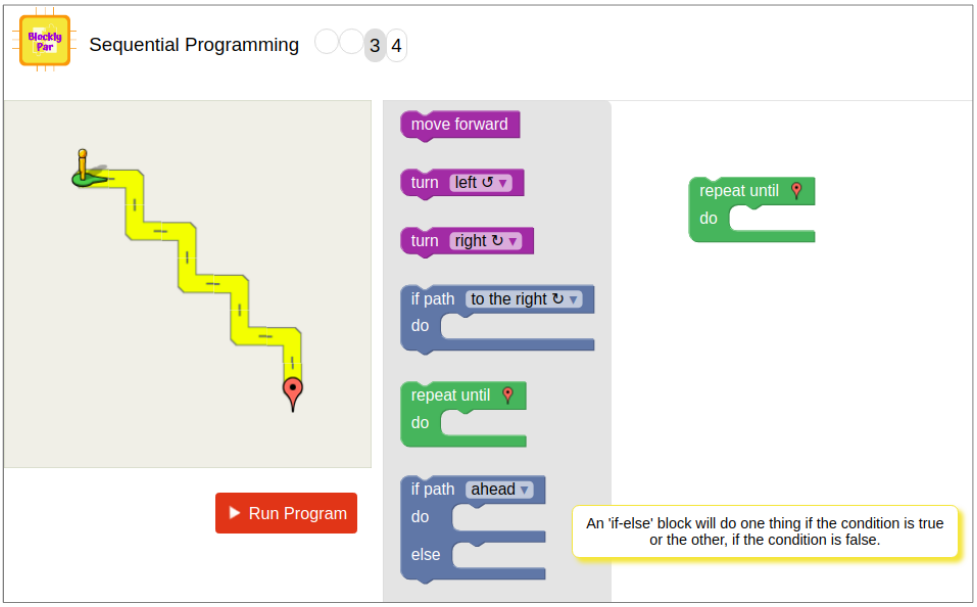


Figura 6.3 – Fase 3 do jogo para introdução à programação com blocos.

6.2 JOGO 2: INTRODUÇÃO A TAREFAS

O segundo jogo apresenta o conceito de execução de tarefas. O desafio deste jogo é orientar o personagem, agora chamado de estudante, a percorrer o caminho até chegar na sala de aula. Porém, para poder ir para a sala de aula e terminar a fase com sucesso, ele deve realizar uma tarefa.

A tarefa deste jogo é concluir todos os trabalhos de casa antes de ir para a aula. Para expressar esta ideia, é introduzido o bloco de realização de tarefa “repeat task until”, onde todos os blocos associados à ele serão executados como em um bloco de repetição, porém verificando se a tarefa foi satisfeita para concluir a fase com sucesso.

A Figura 6.4 apresenta a segunda fase do jogo, onde pode-se observar que um trabalho é considerado concluído quando o estudante passa por ele durante o percurso, então a imagem é retirada do caminho e adicionada na lista de trabalhos finalizados, espaço à esquerda da tela de execução. Os trabalhos são as imagens de computadores com um trecho de código, representando trabalhos de casa reais de alunos de cursos de computação em disciplinas de programação.

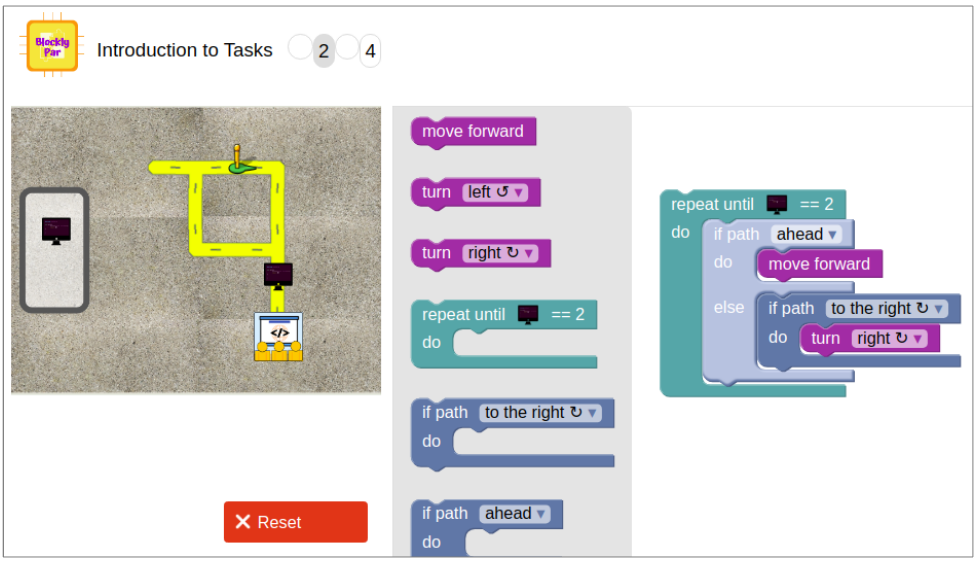


Figura 6.4 – Fase 2 do jogo para introdução a tarefas.

A Figura 6.5 apresenta a última fase do jogo, que introduz o bloco de atribuição de ações a um estudante específico. Agora, para comandar ações sobre o estudante, o usuário deve encaixar os blocos de comando dentro deste bloco, caso contrário nenhuma ação será realizada. Este bloco é apresentado através de um modal inicial no início desta fase.

Em programação paralela, a realização de tarefas utiliza recursos, alocados para executá-las, e lida com uma carga, que são dados envolvidos na execução das tarefas. Buscando associar os atributos utilizados neste jogo com conceitos de programação paralela, definiram-se quais foram os recursos, as tarefas e a carga utilizadas:

- **Recursos:** Estudantes. Um ou dois disponíveis, dependendo da fase.
- **Tarefa:** Terminar todos os trabalhos antes de ir para a aula.
- **Carga:** Trabalhos a serem concluídos antes de chegar na sala de aula.

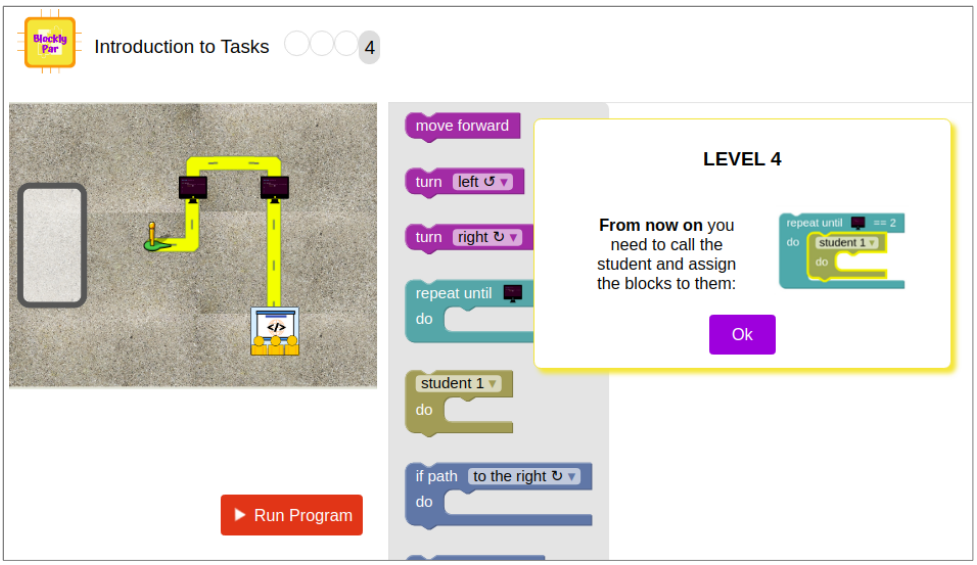


Figura 6.5 – Fase 4 do jogo para introdução a tarefas.

6.3 JOGO 3: PROGRAMAÇÃO PARALELA

O terceiro jogo aborda a programação paralela. O cenário agora é um caminho até a biblioteca, como visto na Figura 6.7. O desafio a ser concluído em cada fase é orientar o estudante até a biblioteca para ele poder devolver todos os livros que estão consigo, representados na lista à esquerda na tela de execução.

Para isso, o usuário deve montar o programa dentro do bloco de tarefas, que realizará as ações repetidamente até todos os livros serem entregues. O estudante entrega um livro ao chegar na biblioteca. Um livro é decrementado da lista, e o estudante retorna automaticamente ao ponto inicial do jogo, onde o bloco de tarefas realizará novamente o percurso definido pelos blocos.

A Figura 6.6 apresenta a primeira fase do jogo, onde ainda não é introduzido o paralelismo. Nesta fase, buscou-se apresentar o cenário e retomar o uso do bloco de atribuição de tarefas a um estudante, apresentado na última fase do jogo anterior, para reforçar o seu uso obrigatório em todas as fases seguintes.

A partir da segunda fase o cenário apresenta mais de um estudante, como visto na Figura 6.7. É introduzido, o bloco de execução de tarefas em paralelo, que possui uma nova cor e nova descrição. O novo bloco é apresentado pelo modal no início da fase e, caso o usuário continue utilizando apenas um estudante para execução sequencial, um *pop-up* aparece relembrando o usuário de usar ambos estudantes. Para selecionar o estudante que irá executar as ações, o usuário deve utilizar o *dropdown* do bloco.

Visando expressar o desempenho adquirido usando mais de um estudante para realizar a ação, o jogo introduz o contador abaixo da tela de execução. Esse recurso é comum em programação paralela, para observar melhorias no desempenho de programas que utilizam mais de um recurso para executar tarefas em paralelo.

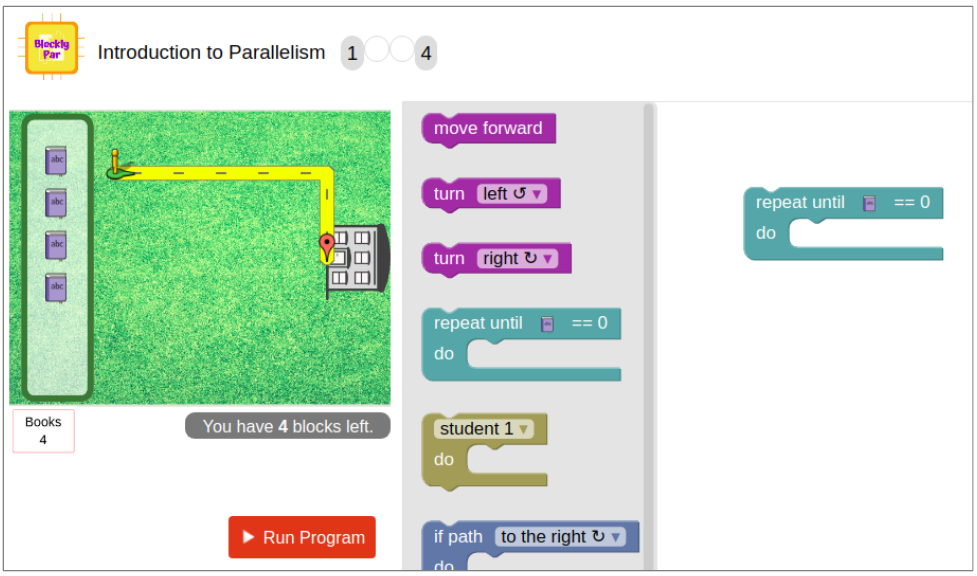


Figura 6.6 – Fase 1 do jogo para introdução à programação paralela.

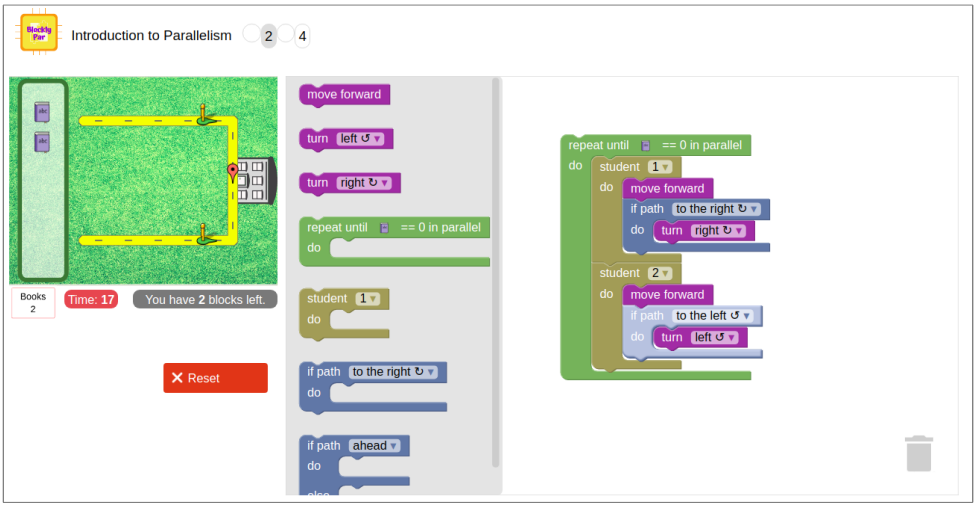


Figura 6.7 – Fase 2 do jogo para introdução à programação paralela.

Com esse jogo, o usuário poderia perceber que a mesma tarefa pode ser executada mais rapidamente se for dividida entre mais de um estudante. Porém, não necessariamente o tempo de execução diminui linearmente utilizando mais de um recurso, pois cada recurso pode executar tarefas com custos diferentes. Na fase 2 em comparação à fase 1, por exemplo, o caminho realizado pelos dois estudantes pode ter o mesmo custo, visto que a distância até a biblioteca é a mesma, mudando apenas o sentido em que cada estudante chegará à ela.

Já na fase 2 em comparação com a fase 4 (Figura 6.8), o uso de mais estudantes não necessariamente irá resultar em um *speedup* linear, pois os caminhos têm custos distintos para chegar até a biblioteca. Isso fica claro, pois o número de blocos atribuído a cada estudante será diferente. De qualquer forma, geralmente irá se obter ganho de desempenho caso forem mais de um estudante para realizar as tarefas.

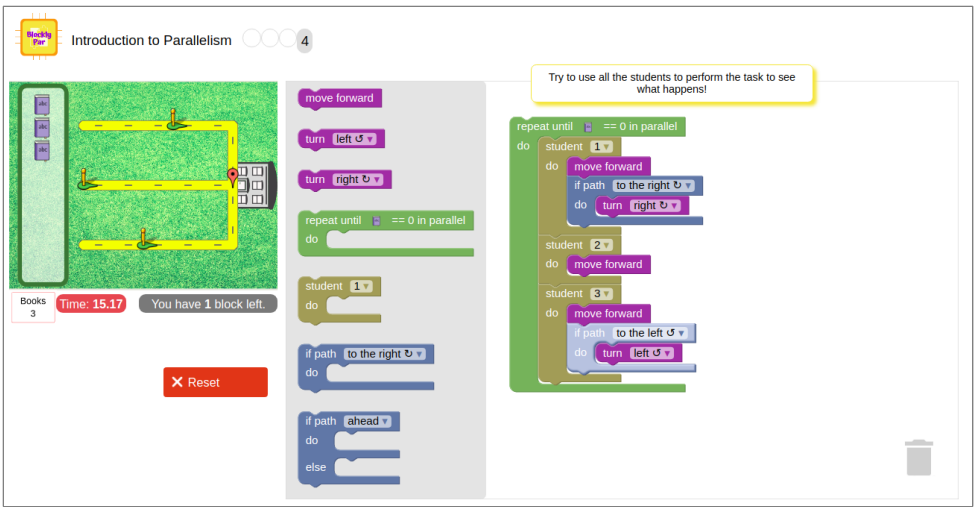


Figura 6.8 – Fase 3 do jogo para introdução à programação paralela.

O contador é essencial nesta fase, pois o usuário começa a adquirir a noção de otimização da sua execução ao utilizar paralelismo. No modal final é apresentado o tempo de execução gasto nas fases, e em alguns modais iniciais é relembrado ao usuário para prestar atenção no tempo de execução utilizado. Os atributos utilizados neste jogo relacionados à programação paralela foram:

- **Recursos:** Estudantes. De um à três disponíveis.
- **Tarefa:** Entregar todos os livros da lista na biblioteca.
- **Carga:** Livros a serem entregues na biblioteca.
- **Métrica de desempenho:** Tempo de execução gasto na realização da tarefa.

7 TESTES COM USUÁRIOS

Os testes e avaliações foram realizados com 13 alunos de graduação e pós-graduação em Ciência da Computação e Sistemas de Informação da Universidade Federal de Santa Maria. Foram selecionados dois públicos: (i) alunos com conhecimento aprofundado de programação paralela, que já utilizaram programação paralela em trabalhos não relacionados a disciplinas da grade curricular, (ii) e alunos com conhecimento básico ou sem conhecimento de programação paralela, de qualquer semestre dos cursos.

Foram criados 2 formulários avaliativos, um para cada público, no Google Form[1], uma ferramenta da Google para criação de formulários personalizados. Essa ferramenta foi escolhida visto a facilidade em compartilhar o formulário entre os usuários, podendo restringir o acesso aos usuários com e-mail dos cursos de Computação e devido a opção de exportar as respostas para uma planilha.

Os formulários continham questões de múltipla-escolha e dissertativas, com 14 questões para os alunos de pós-graduação e 12 para os de graduação. As primeiras questões foram iguais para ambos, perguntando qual o navegador utilizado, se o usuário conhecia o Blockly-Games, se já havia jogado o Maze e se já cursou a disciplina de Programação Paralela. Ao final, ambos podiam escrever críticas e sugestões.

A diferença entre os formulários estava nas seções intermediárias, onde os alunos de pós-graduação responderam questões sobre a usabilidade da plataforma, a assimilação de conceitos de paralelismo com os blocos criados e opinião sobre os textos nos modais. Os alunos de graduação responderam questões mais voltadas ao aprendizado de paralelismo após as jogadas e qual a opinião sobre o público mais apropriado para o BlocklyPar.

Buscando obter informações no contexto de usabilidade, a primeira etapa dos testes foi realizada com os especialistas em programação paralela. Com os resultados desses testes foram observadas melhorias a serem corrigidas nos jogos e a sugestão para a implementação de um jogo introdutório, realizada antes da aplicação para os alunos de graduação.

7.1 ESPECIALISTAS EM PROGRAMAÇÃO PARALELA

Os testes com esse público foram realizados à distância, através de um formulário com perguntas a serem respondidas antes e após o uso da plataforma. Participaram 2 alunos do mestrado em Ciência da Computação que cursaram a disciplina de Programação Paralela no 9° e no 3° semestre do curso de graduação em Ciência da Computação, e que utilizam abordagens paralelas em seus trabalhos de mestrado.

[1] https://www.google.com/forms/about/

Os alunos testaram apenas o jogo 2 e 3 da plataforma, visto que o jogo 1 de introdução à programação com blocos ainda não havia sido implementado. Ambos alunos relataram que conseguiram utilizar bem todos os blocos do jogo, porém, deve-se ressaltar que eles já conheciam o Blockly-Games e já haviam jogado o jogo Maze. Os dois aprovaram as descrições apresentadas nos modais de início das fases:

> *"Bem claros e concisos"*
> *"Achei objetivos e úteis, pois ressaltam o que é importante ser observado na fase de uma forma bastante compreensível e clara."*

Um dos alunos ressaltou que o fluxo de surgimento dos blocos foi bom, mas partindo do princípio de que quem for utilizar o BlockyPar já tenha conhecimento sobre os blocos lógicos básicos. Assim, para os testes com os usuários iniciantes em programação paralela, foi implementado o Jogo 1, visando familiarizar o usuário com a plataforma de programação com blocos.

Além disso, os alunos levantaram sugestões e melhorias. Um dos alunos relatou problemas na fase 4, onde os livros não foram consumidos pelo estudante ao atingir a biblioteca, e assim o programa não terminava. O problema foi observado no navegador Google Chrome ao acessar pelo celular e no Firefox ao acessar no computador.

Como sugestões, um aluno relatou que seria interessante armazenar o histórico dos tempos de execução em cada fase para comparar o quão mais rápida foi a execução paralela. Outra sugestão foi a de disponibilizar um bloco para executar todas as ações para todos os estudantes ao mesmo tempo, em paralelo.

7.2 FORMULÁRIO PARA OS INICIANTES EM PROGRAMAÇÃO PARALELA

O formulário avaliativo para os alunos iniciantes em programação paralela, graduandos de cursos de Computação, foi aplicado após os usuários jogarem os três jogos do BlocklyPar. O formulário foi separado em 4 seções:

- **Seção 1 - Informações do usuário:** Pergunta sobre o curso do usuário, o semestre em que está estudando e o navegador utilizado.
- **Seção 2 - Antes do jogo:** Verificação se o usuário já conhecia o Blockly-Games e se já havia jogado o jogo Maze da plataforma; Pergunta se o usuário já cursou a disciplina "ELC139 - Programação Paralela", ofertada como uma Disciplina Complementar de Graduação na universidade dos usuários, e em qual semestre a cursou.

- **Seção 3 - Sobre programação paralela:** Nesta seção foram criadas 4 perguntas que envolvessem conceitos de programação paralela relacionados aos jogos do BlocklyPar. As perguntas buscaram obter informações sobre a eficiência da plataforma em abstrair conceitos paralelos através dos desafios propostos e dos novos blocos criados.

  – **Pergunta 1:** Buscou-se verificar se o usuário conseguiu entender o conceito de realização de tarefas e os blocos de programação associados a este conceito. A Figura 7.1 apresenta a pergunta e as opção disponíveis para o usuário marcar, podendo selecionar quantas quisesse. Esperava-se que os alunos marcassem as Opções 1, 2 e 3, que mostram os blocos de realização de tarefas de forma sequencial e paralela. A opção 4 mostra o bloco de atribuição da tarefa ao usuário, que apenas redireciona as ações ao usuário selecionado no *dropdown* do bloco.

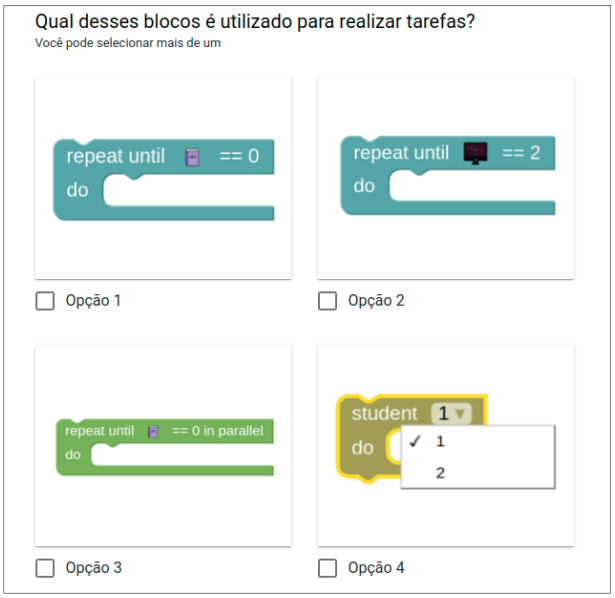


Figura 7.1 – Primeira pergunta da seção de programação paralela.

  – **Pergunta 2:** Esta pergunta, apresentada na Figura 7.2, serviu para verificar se o usuário compreendeu a ideia de execução de tarefas, independentemente de saber qual bloco determina esta ação. Foi apresentada uma imagem de uma situação relacionada ao cenário do Jogo 3, e o questionamento sobre qual recurso do jogo representa os dados do problema. A resposta correta é a C, de que os dados são os livros, que o estudante deve levar até a biblioteca. O usuário seria o recurso alocado para a execução da tarefa, e os passos seriam os resultados da execução das instruções determinadas pelos blocos de movimento.

  – **Pergunta 3:** Esta pergunta verificou se um bloco novo foi associado corretamente com a estratégia de programação paralela que ele representa. Buscou-

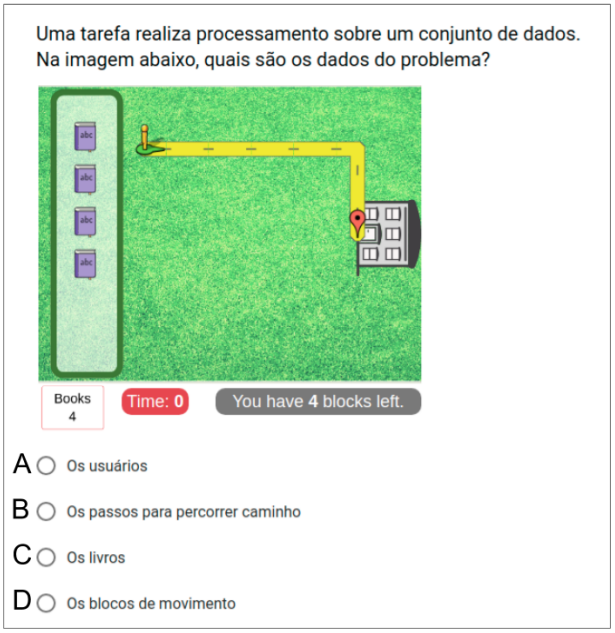


Figura 7.2 – Segunda pergunta da seção de programação paralela.

se obter a compreensão dos usuários sobre a utilização do bloco de atribuição de tarefa a um estudante, para saber com qual abordagem de programação paralela ele poderia ser comparado. A Figura 7.3 apresenta a pergunta, com a imagem do bloco de estudante destacado. A resposta correta é a segunda opção (B), de que o bloco representa a associação de uma tarefa a um processador específico, no caso, associando a tarefa ao estudante selecionado.

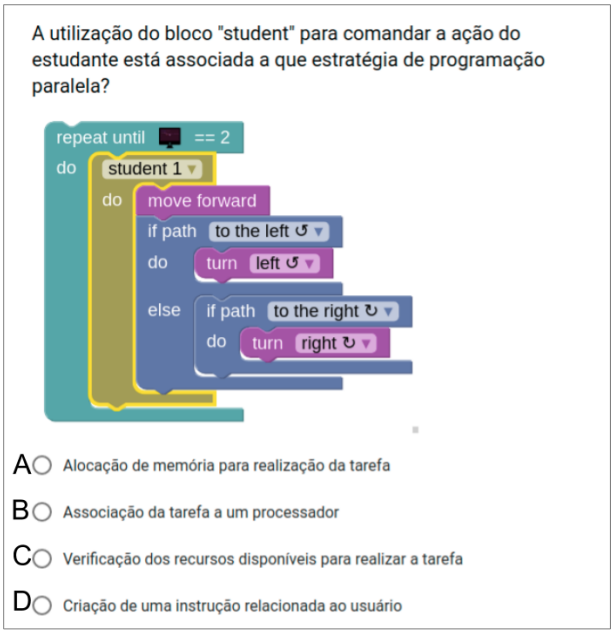


Figura 7.3 – Terceira pergunta da seção de programação paralela.

– **Pergunta 4:** Por fim, a quarta pergunta (Figura 7.4) serviu para obter a compreensão dos alunos sobre análise de desempenho. Foi apresentada uma situação

relacionada aos objetivos do Jogo 3, onde um estudante deveria entregar 8 livros na biblioteca. A pergunta era qual das outras situações apresentadas nas opções levaria a metade do tempo de execução para entregar todos os livros em comparação com a situação apresentada na pergunta.

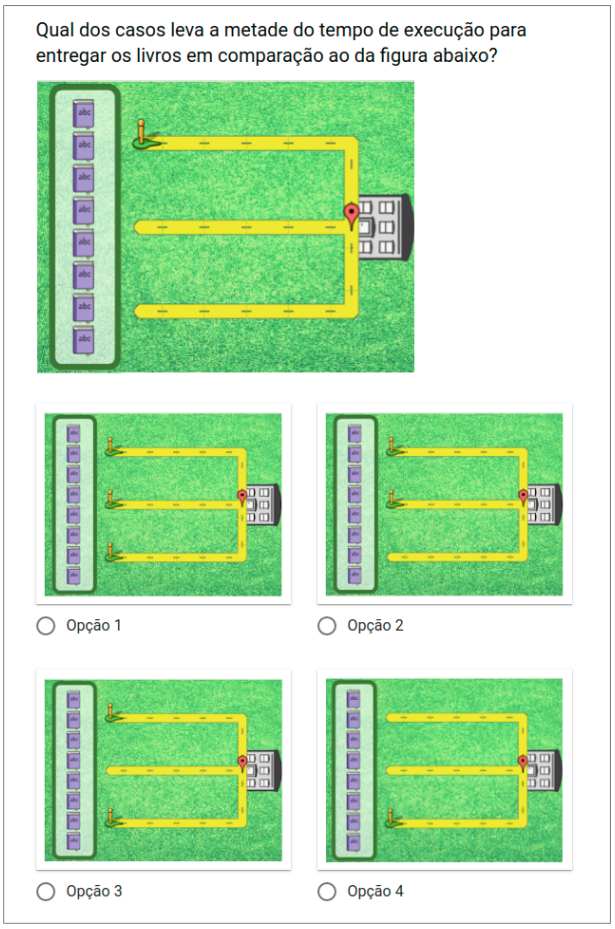


Figura 7.4 – Quarta pergunta da seção de programação paralela.

A resposta correta para essa questão era a Opção 3, onde são utilizados dois estudantes para realizar a tarefa, e ambos percorrerão o mesmo número de passos até chegar na biblioteca. As outras opções, ou resultariam em um tempo de execução menor do que a metade do tempo em relação ao apresentado, ou o mesmo tempo como na opção 4. Com essa pergunta também buscava-se compreender se o usuário prestou atenção no contador de tempo de execução nas fases do Jogo 3.

- **Seção 4 - Sobre o BlocklyPar:**

  A primeira pergunta investigou qual seria o público mais apropriado para utilizar o BlocklyPar, podendo marcar mais de uma resposta: (i) alunos de ensino fundamental, (ii) alunos de ensino médio, (iii) calouros de cursos de Computação com conhecimento ou não de programação paralela, (iv) alunos de ensino superior em Computa-

ção sem conhecimento de programação paralela, (v) alunos de ensino superior em Computação com conhecimento de programação paralela.

A segunda pergunta buscou obter quais conceitos de programação paralela o usuário conseguiu associar com os jogos do BlocklyPar, caso tenha associado algum. Caso não tivesse conhecimento de programação paralela o esperado era que o usuário deixasse a resposta em branco. E a última pergunta foi um espaço de texto para sugestões e melhorias na plataforma.

7.3 INICIANTES EM PROGRAMAÇÃO PARALELA

Os testes com alunos iniciantes em programação paralela foram feitos presencialmente com 8 dos 11 participantes totais. Cada usuário teve acesso a um computador para jogar os 3 jogos em um intervalo de tempo reservado de 30 minutos, definido pelas jogadas realizadas, onde nas fases do primeiro jogo se gastava um máximo de 2 minutos cada e nas dos jogos seguintes cerca de 3 minutos.

Nenhum dos alunos tomou mais tempo do que o determinado, e sabendo que haveria um formulário a ser respondido posteriormente, foi disponibilizado um bloco de anotações e uma caneta para cada um poder anotar críticas e erros encontrados. O formulário avaliativo utilizado está disponível em: https://forms.gle/hiwGX38hgStXACR9A.

A orientação aos usuários era de ler a página “About” antes de iniciar os jogos, mas sem obrigatoriedade dessa ação, e após resolver os três jogos propostos. Visto que a plataforma estava em inglês e nem todos usuários tem conhecimento da língua, permitiu-se a solicitação de tradução dos textos pelo avaliador. A maioria dos usuários conhecia o Blockly-Games, mas não necessariamente haviam jogado o jogo Maze da plataforma, conforme visto nos gráficos de resposta na Figura 7.5.

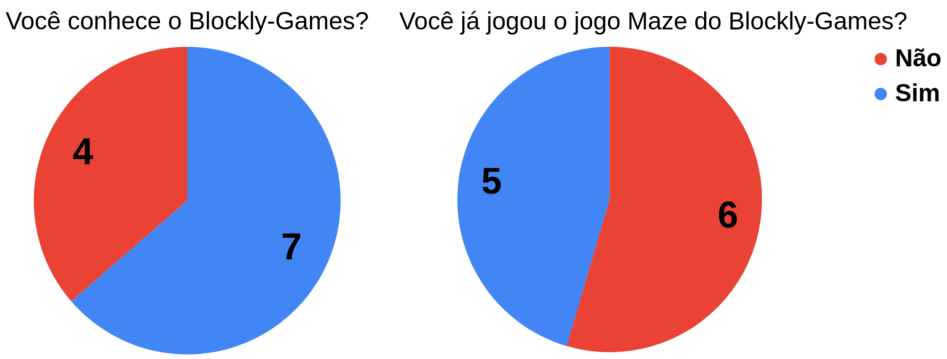


Figura 7.5 – Conhecimento dos usuários sobre o Blockly-Games.

Os usuários foram 2 alunos do curso de Sistemas de Informação e 9 de Ciência da Computação, sendo 6 meninas e 5 meninos. Os alunos estão em semestres variados, de

2° à 12° semestre de estudos, e 4 já cursaram a disciplina de Programação Paralela. Vale ressaltar que essa disciplina pode ser realizada em ambos os cursos.

Na Tabela 7.1 são apresentadas as respostas corretas para cada questão, as respostas mais marcadas sobre programação paralela e quantos alunos as marcaram entre parênteses. Considerou-se que a maioria dos usuários não tinha conhecimento sobre programação paralela, e portanto poderia ter mais dificuldade para responder as questões.

Questão	Resposta correta	1^a mais marcada	2^a mais marcada	3^a mais marcada
1	Opções 1, 2 e 3	Opção 2 (7)	Opção 4 (6)	Opção 3 (4)
2	C	C (7)	B (2)	D (2)
3	B	B (9)	D (2)	-
4	Opção 3	Opção 3 (7)	Opção 2 (3)	Opção 1 (1)

Tabela 7.1 – Respostas dos usuários às perguntas da seção 3 do formulário. Alternativa mais marcada e número de usuários que a marcaram entre parênteses.

Com os resultados da seção 3, pôde-se observar que a maioria dos alunos acertou as questões propostas, tendo ou não conhecimento anterior de programação paralela. Destaca-se que 6 usuários marcaram a opção 4 na primeira questão, sobre quais blocos eram responsáveis pela realização de tarefas, quando o bloco da opção 4 era o único que não era associado a tarefas.

Outra observação é em relação à questão 4 (Figura 6.8), onde 3 usuários selecionaram a opção 2 como resposta. Esta opção traz a utilização de dois estudantes conforme a resposta correta, porém o seu tempo de execução não será exatamente a metade do tempo gasto com um estudante, mas será menor, pois um dos estudantes percorre um caminho mais rápido.

O público-alvo do BlocklyPar são alunos que estão nos primeiros semestre do ensino superior em cursos de computação. A Figura 7.6 apresenta as respostas para a pergunta que verifica a opinião dos usuários sobre o público adequado para a plataforma, onde o usuário podia selecionar quantas opções quisesse. A mais selecionada foi a de que o público adequado são calouros em cursos de Computação com ou sem conhecimento de programação paralela, seguida de alunos de ensino superior em cursos de Computação sem conhecimento de programação paralela.

Para a pergunta sobre os conceitos de paralelismo abstraídos com o BlocklyPar, 2 alunos dos 4 que já cursaram Programação Paralela relataram os seguintes conceitos: divisão de tarefas, alocação e compartilhamento de recursos, acesso simultâneo/concorrente aos dados, e de que o estudante representaria uma *thread*.

A última pergunta era uma caixa de texto livre para críticas e sugestões. Todos os 9 usuários que responderam este campo citaram gostar do jogo e aprovar a proposta, relatando, por exemplo:

“Achei o jogo interessante e instiga o raciocínio, principalmente porque pro-

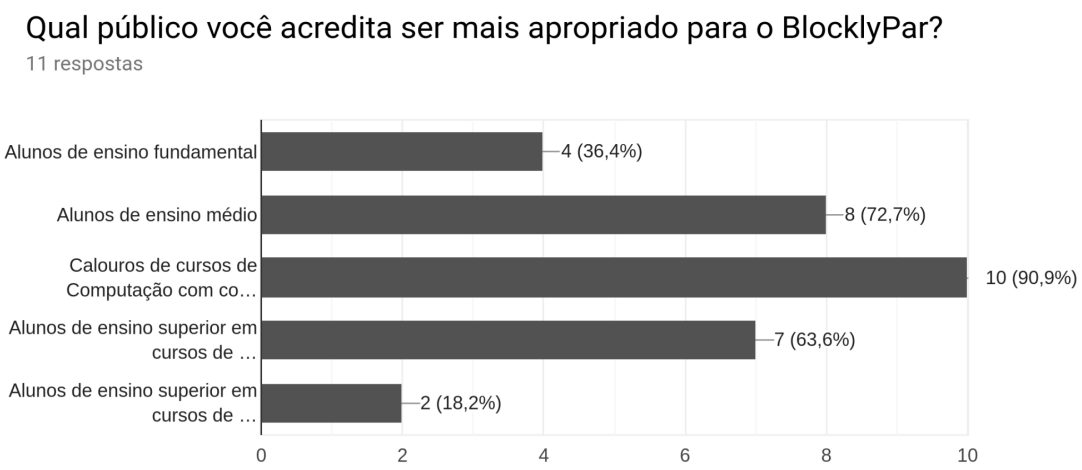


Figura 7.6 – Respostas para a primeira pergunta da seção 4.

> *põe exercícios com conceitos de programação paralela (não tão explorados em jogos similares já existentes).”*
> *“O jogo traz uma experiência diversificada, ajudando a atribuir conhecimento mesmo sem o conhecimento teórico.”*
> *“Gostei muito da ideia e da iniciativa, a mesma é super interessante principalmente para iniciantes da área!”*

Além disso, alguns usuários explicaram melhor sobre qual público eles consideram apropriado para o BlocklyPar:

> *“Se aplicado com crianças, acredito que seria interessante ter alguma atividade que representasse alguma ação cotidiana, visando instigá-las a relacionar e entender que os conceitos de programação/computação podem ajudá-las a resolver seus problemas do dia-a-dia.”*
> *“Acho que o poderia ser usado com alunos mais novos (ensino fundamental e médio), mas só se eles souberem conceitos de programação, não necessariamente de paralela.”*
> *“Acredito que possa ser aplicado tanto para quem possui ou não conhecimentos em programação (em geral). Porém, para quem não tem conhecimento em programação, seria interessante aplicar como sequência de outras atividades mais introdutórias.”*

Foram coletadas sugestões e melhorias sobre o BlocklyPar, incluindo erros encontrados, sugestões de mudanças nos recursos existentes e introdução de novos recursos. Nessas respostas, foram identificados aspectos a serem melhorados no Blockly, que são

utilizados no Blockly-Games por exemplo, e quanto as novas funcionalidades apresentadas pelo BlocklyPar.

Críticas e sugestões relacionadas à funcionalidade da biblioteca Blockly:

- Opção de remover apenas um bloco no meio de um conjunto de blocos encaixados, sem os outros blocos serem arrastados junto com a trilha;
- Opção do modal inicial ser aberto novamente durante a partida, com um botão de "Ajuda", por exemplo;
- Um sinal ou bloco que aponte o início do algoritmo na tela de programação;
- Quando o boneco colidir com a parede, e portanto ficar trancado na fase, poderia se determinar um tempo limite para ele ficar trancado, e após reiniciar a fase automaticamente.

Críticas e sugestões relacionadas a funcionalidade do BlocklyPar:

- Uma apresentação inicial sobre cada um dos blocos e o seu funcionamento, como uma documentação no próprio site;
- Apresentação mais clara sobre o objetivo de cada jogo, pois não é intuitivo que a tarefa é orientar o personagem a percorrer um caminho usando os blocos. No jogo 3, por exemplo, não fica claro que o estudante retorna ao seu local de origem após entregar um livro;
- A apresentação do bloco "student" na última fase do Jogo 2 trouxe confusão aos usuários, visto que eles recém haviam adquirido conhecimento com um conceito novo, de execução de tarefas;
- Apresentar um identificador do número de cada estudante, para relacionar ao bloco de estudantes;
- Criar caminhos mais diversificados para variar os blocos que serão utilizados, ou forçar a utilização de outros blocos, contribuindo para o aprendizado do usuário sobre a ação de cada bloco;
- Criar mais níveis de programação paralela para fixar melhor os conceitos;
- Deixar mais claro a medida dos tempos de execução e o que eles representam;
- Erros encontrados: as tarefas do Jogo 2 podiam ser coletados mais de uma vez caso o usuário passasse novamente pelo mesmo caminho, e dependendo do bloco usado, os livros do Jogo 3 não sumiam da lista mas o jogo contabilizava que eles eram entregues, finalizando a fase normalmente.

7.4 COMPORTAMENTO DOS USUÁRIOS

Durante os testes presenciais, foram observados comportamento semelhantes em alguns usuários. Três alunos apresentaram dificuldade com orientação em diferenciar direita e esquerda e por isso demoraram mais tempo para resolver os desafios do que outros que não conseguiram.

Por ser em inglês, três alunos tiveram dificuldade em entender o objetivo em cada fase descrito nos modais. Ainda sobre os modais, alguns alunos leram os textos e outros não, sendo que os que não leram sentiram a falta deles depois, tendo que recarregar a página para os ver novamente e entender o desafio.

Além disso, observou-se que os alunos que já haviam jogado o Maze do BlocklyGames resolveram o jogo mais rapidamente. Os que não conheciam o jogo mas que relataram experiência com jogos semelhantes, ou estão acostumados a jogar, descobriram rapidamente o fluxo de arraste e encaixe de blocos e o modo como se deveria compor a resposta. Um dos alunos que não conhecia ferramentas com blocos e não jogava, teve dificuldade logo na primeira fase, de entender o objetivo do jogo e de como orientar um personagem até um ponto final usando arraste e encaixe.

Utilizando o Google Analytics, pode-se obter informações gerais sobre o acesso ao BlocklyPar. Apenas 8 dos 13 alunos acessaram a página de “About” conforme sugerido, ficando cerca de 2 minutos e meio de tempo médio na página. Os usuários que acessaram essa página não tiveram taxa de rejeição, ou seja, eles não abriram a página e a abandonaram logo em seguida.

No segundo jogo, a última fase foi observada como a que os usuários levaram mais tempo para resolver, principalmente devido à inserção do novo bloco para definir o estudante. Com o Google Analytics, pode-se confirmar essa observação visto que o tempo médio de acesso a esta fase foi de 2 minutos e 47 segundos, maior do que o tempo para resolução das outras.

Já no terceiro jogo, a primeira fase levou mais tempo para ser executada do que a primeira fase do jogo 2. Supõe-se que isso tenha ocorrido devido a novidade do cenário e do objetivo do jogo, e a retomada ao uso do bloco para definir o estudante, visto que nesta fase houveram usuários que tentaram resolvê-las sem usar este bloco.

Todas as fases tiveram tempos médios entre 1 e 2 minutos, com exceção da última fase, que trazia três estudantes para resolver a tarefa, levando um tempo médio de 4 minutos e 36 segundos. Foi observado que os estudantes saiam desta página após concluir o desafio, então o tempo gasto foi realmente para implementar a resposta.

8 RESULTADOS

A partir dos testes e avaliações com usuários, foram implementadas melhorias em aspectos visuais e funcionais do jogo. Os modais receberam menos texto e mais explicações sobre o objetivo de cada fase, os erros observados quanto a imagem do computador e dos livros foram corrigidos, e demais melhorias nas imagens e cores do jogo foram implementadas.

Observou-se que a ideia de análise de desempenho não conseguiu ser bem compreendida, devido às respostas do formulário dos alunos iniciantes em programação paralela, na questão 3 da seção 3. Os usuários perceberam que a execução ficou mais rápida ao se utilizar mais de um estudante para realizar a ação, mas não compararam o tempo gasto entre as fases para entender o quão mais rápida foi essa melhoria.

Nem todos os usuários identificaram quais eram os blocos de tarefas, confundindo o bloco de atribuição a um estudante como o bloco que realiza as tarefas, conforme visto na questão 1 da seção 3 do segundo formulário. Para melhorar essa percepção, sugere-se alterar a cor, texto e/ou forma desses blocos, buscando diferenciá-los e destacá-los dos demais, para que sejam mais compreensíveis ao usuário.

A criação de um bloco que movimente todos os estudantes ao mesmo tempo parece ser uma proposta eficaz para a assimilação de paralelismo. Com este bloco, o usuário poderá observar que mesmo lidando com os mesmos dados, nem sempre os recursos alocados executarão as mesmas instruções para realizar suas tarefas. No contexto do jogo, os estudantes podem ter que percorrer caminhos diferentes para atingir um mesmo objetivo.

O público-alvo mais escolhido pelos usuários foi o público determinado neste projeto: alunos ingressantes em cursos superiores de Computação. Acredita-se portanto que o BlocklyPar tenha potencial para instigar o pensamento paralelo e introduzir conceitos de programação paralela a estudantes com conhecimento básico de programação de computadores, mesmo que saibam apenas programar com linguagem textual.

9 CONCLUSÃO

Computadores modernos possuem arquiteturas *multicore* que podem ser exploradas para a execução de mais de uma tarefa ao mesmo tempo, em paralelo. Computação de Alto Desempenho é uma área da computação que ensina conceitos, abordagens e ferramentas para implementar programas paralelos, obtendo melhor desempenho nesses ambientes em comparação a programas seriais.

A programação paralela vem sendo cada vez mais utilizada em computação, pois permite, por exemplo, diminuir o tempo de execução de programas que processam grandes quantidades de dados. Inclusive, muitos desenvolvedores consideram o dispositivo de execução alvo antes de implementar o seu algoritmo, impactando nas decisões sobre a lógica e sobre as estruturas de dados utilizadas.

Em cursos superiores de computação, disciplinas de programação paralela costumam ser ofertadas como optativas, após o aluno adquirir experiência com a programação serial. Já para iniciantes na área, os jogos disponíveis para ensino de programação abordam conceitos lógicos e estruturas de dados utilizando apenas instruções sequenciais. Com isso, programadores aprendem a implementar algoritmos seriais sem considerar o uso de paralelismo para otimizar a execução de seus programas.

Este trabalho apresentou BlocklyPar[1], uma nova proposta de plataforma de jogos para introdução ao pensamento paralelo utilizando programação visual com blocos. BlocklyPar é voltada principalmente a estudantes iniciantes em cursos de graduação em computação, e por isso traz cenários e atividades do dia-a-dia de universitários, mostrando que até mesmo tarefas corriqueiras podem se beneficiar de abordagens paralelas.

Os jogos foram inspirados na plataforma Blockly-Games, que visa ensinar programação através de uma série de jogos educacionais. Essa e outras plataformas de programação visual utilizam a biblioteca Blockly para criação dos blocos a partir de linguagem textual. O Blockly disponibiliza uma biblioteca de blocos prontos que tratam apenas programação sequencial. Por isso, além do projeto do BlocklyPar, tiveram que ser criados novos blocos para expressar o paralelismo a nível de usuário, abordando conceitos básicos de programação paralela e introduzindo termos utilizados na área.

A partir dos testes realizados, foram levantadas sugestões de melhorias na plataforma. Como trabalhos futuros, espera-se criar mais fases para os jogos, criar novos blocos para abordar outros conceitos de programação paralela, fazer a tradução para português brasileiro e disponibilizar uma documentação detalhada sobre o objetivo de cada jogo e funcionalidade de cada bloco.

Espera-se que a plataforma BlocklyPar seja utilizada como recurso didático para a introdução ao pensamento paralelo ao mesmo tempo em que aborda o ensino-aprendizagem

[1] https://blocklypar.github.io/

de programação de computadores. Também, espera-se que os usuários adquiram noções sobre o fluxo de execução de programas paralelos em ambientes *multicore*, e em como o bom uso de recursos computacionais influencia no desempenho de seus programas. Por fim, espera-se que os desafios do BlocklyPar instiguem a curiosidade dos usuários em aplicar programação paralela naturalmente em diferentes áreas da computação.

O código-fonte da plataforma está disponível em: https://github.com/blocklypar. E o BlocklyPar em: https://blocklypar.github.io/.

REFERÊNCIAS BIBLIOGRÁFICAS

ARMONI, M.; MEERBAUM-SALANT, O.; BEN-ARI, M. From scratch to “real” programming. **Trans. Comput. Educ.**, ACM, New York, NY, USA, v. 14, n. 4, p. 25:1–25:15, fev. 2015. ISSN 1946-6226. Disponível em: <http://doi.acm.org/10.1145/2677087>.

BAU, D. et al. Learnable programming: Blocks and beyond. **Commun. ACM**, ACM, New York, NY, USA, v. 60, n. 6, p. 72–80, maio 2017. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/3015455>.

BURKE, K. Chapel: A versatile language for teaching parallel programming: Conference workshop. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 30, n. 6, p. 16–16, jun. 2015. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=2753024.2753030>.

CHAPMAN, B.; JOST, G.; PAS, R. van der. **Using OpenMP: Portable Shared Memory Parallel Programming.** USA: The MIT Press, 2007. ISBN 978-0-262-53302-7.

Feng, A.; Feng, W. Parallel programming with pictures in a snap! In: **2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2016. p. 950–957.

FENG, A.; GARDNER, M.; FENG, W.-c. Parallel programming with pictures is a snap! **J. Parallel Distrib. Comput.**, Academic Press, Inc., Orlando, FL, USA, v. 105, n. C, p. 150–162, jul. 2017. ISSN 0743-7315. Disponível em: <https://doi.org/10.1016/j.jpdc.2017.01.018>.

FENG, A.; TILEVICH, E.; FENG, W.-c. Block-based programming abstractions for explicit parallel computing. In: **Proceedings of the 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)**. Washington, DC, USA: IEEE Computer Society, 2015. (BLOCKS AND BEYOND ’15), p. 71–75. ISBN 978-1-4673-8367-7. Disponível em: <http://dx.doi.org/10.1109/BLOCKS.2015.7369006>.

FOSTER, I. **Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0201575949.

GARDNER, W. B. Should we be teaching parallel programming? In: **Proceedings of the 22Nd Western Canadian Conference on Computing Education**. New York, NY, USA: ACM, 2017. (WCCCE ’17), p. 3:1–3:7. ISBN 978-1-4503-5066-2. Disponível em: <http://doi.acm.org/10.1145/3085585.3085588>.

GILAT, A. **MATLAB: An Introduction with Applications 2nd Edition**. [S.l.]: John Wiley Sons, 2004. ISBN 978-0-471-69420-5.

GREGG, C. et al. Ecosim: A language and experience teaching parallel programming in elementary school. In: **Proceedings of the 43rd ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2012. (SIGCSE ’12), p. 51–56. ISBN 978-1-4503-1098-7. Disponível em: <http://doi.acm.org/10.1145/2157136.2157155>.

IEEE. Standard for information technology–portable operating system interface (posix(r)) base specifications, issue 7. **IEEE Std 1003.1, 2016 Edition (incorporates IEEE Std 1003.1-2008, IEEE Std 1003.1-2008/Cor 1-2013, and IEEE Std 1003.1-2008/Cor 2-2016)**, p. 1–3957, Sep. 2016.

KELLEHER, C.; PAUSCH, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 37, n. 2, p. 83–137, jun. 2005. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/1089733.1089734>.

MASCARELL, J. B. Visual help to learn programming. **ACM Inroads**, ACM, New York, NY, USA, v. 2, n. 4, p. 42–48, dez. 2011. ISSN 2153-2184. Disponível em: <http://doi.acm.org/10.1145/2038876.2038891>.

MELLOR-CRUMMEY, J.; GROPP, W.; HERLIHY, M. Teaching parallel programming: A roundtable discussion. **XRDS**, ACM, New York, NY, USA, v. 17, n. 1, p. 28–30, set. 2010. ISSN 1528-4972. Disponível em: <http://doi.acm.org/10.1145/1836543.1836553>.

MOSKAL, B.; LURIE, D.; COOPER, S. Evaluating the effectiveness of a new instructional approach. In: **Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2004. (SIGCSE '04), p. 75–79. ISBN 1-58113-798-2. Disponível em: <http://doi.acm.org/10.1145/971300.971328>.

NEZU, N. Teaching parallel programming in 50 minutes. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 31, n. 2, p. 18–24, dez. 2015. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=2831432.2831435>.

ONTANON, S. et al. Designing visual metaphors for an educational game for parallel programming. In: **Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems**. New York, NY, USA: ACM, 2017. (CHI EA '17), p. 2818–2824. ISBN 978-1-4503-4656-6. Disponível em: <http://doi.acm.org/10.1145/3027063.3053253>.

PACHECO, P. S. **Parallel Programming with MPI**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996. ISBN 1-55860-339-5.

PAPERT, S. Teaching children thinking. **Programmed Learning and Educational Technology**, Routledge, v. 9, n. 5, p. 245–255, 1972. Disponível em: <https://doi.org/10.1080/1355800720090503>.

PASTERNAK, E.; FENICHEL, R.; MARSHALL, A. N. Tips for creating a block language with blockly. In: . [S.l.: s.n.], 2017. p. 21–24.

Pasternak, E.; Fenichel, R.; Marshall, A. N. Tips for creating a block language with blockly. In: **2017 IEEE Blocks and Beyond Workshop (B B)**. [S.l.: s.n.], 2017. p. 21–24.

PIMENTEL, M.; FILIPPO, D.; SANTORO, F. M. Design science research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação. In: PORTO ALEGRE (RS). Sociedade Brasileira de Computação. **Metodologia de Pesquisa em Informática na Educação: Concepção da Pesquisa.** Porto Alegre. v. 1, cap. 5. Disponível em: <https://metodologia.ceie-br.org/livro-1/>. Acesso em: 6 agosto 2019.

RIZVI, M. et al. A cs0 course using scratch. **J. Comput. Sci. Coll.**, Consortium for Computing Sciences in Colleges, USA, v. 26, n. 3, p. 19–27, jan. 2011. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=1859159.1859166>.

ROY, K. App inventor for android: Report from a summer camp. In: . [S.l.: s.n.], 2012. p. 283–288.

ROY, K.; ROUSSE, W.; DEMERITT, D. Comparing the mobile novice programming environments: App inventor for android vs. gamesalad. In: . [S.l.: s.n.], 2012. p. 1–6. ISBN 978-1-4673-1353-7.

SHAFI, A. et al. Teaching parallel programming using java. In: **Proceedings of the Workshop on Education for High-Performance Computing**. Piscataway, NJ, USA: IEEE Press, 2014. (EduHPC '14), p. 56–63. ISBN 978-1-4799-7021-6. Disponível em: <http://dx.doi.org/10.1109/EduHPC.2014.7>.

SHAPIRO, R. B.; AHRENS, M. Beyond blocks: Syntax and semantics. **Commun. ACM**, ACM, New York, NY, USA, v. 59, n. 5, p. 39–41, abr. 2016. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2903751>.

SHUHIDAN, S. M.; HAMILTON, M.; D'SOUZA, D. Understanding novice programmer difficulties via guided learning. In: **Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education**. New York, NY, USA: ACM, 2011. (ITiCSE '11), p. 213–217. ISBN 978-1-4503-0697-3. Disponível em: <http://doi.acm.org/10.1145/1999747.1999808>.

TUMLIN, N. Teacher configurable coding challenges for block languages. In: **Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2017. (SIGCSE '17), p. 783–784. ISBN 978-1-4503-4698-6. Disponível em: <http://doi.acm.org/10.1145/3017680.3022467>.

VICTOR, B. **LEARNABLE PROGRAMMING Designing a programming system for understanding programs**. 2012. Disponível em: <http://worrydream.com/LearnableProgramming/>.

WEINTROP, D. Minding the gap between blocks-based and text-based programming (abstract only). In: **Proceedings of the 46th ACM Technical Symposium on Computer Science Education**. New York, NY, USA: ACM, 2015. (SIGCSE '15), p. 720–720. ISBN 978-1-4503-2966-8. Disponível em: <http://doi.acm.org/10.1145/2676723.2693622>.

WEINTROP, D.; WILENSKY, U. To block or not to block, that is the question: Students' perceptions of blocks-based programming. In: **Proceedings of the 14th International Conference on Interaction Design and Children**. New York, NY, USA: ACM, 2015. (IDC '15), p. 199–208. ISBN 978-1-4503-3590-4. Disponível em: <http://doi.acm.org/10.1145/2771839.2771860>.