Um Ambiente Baseado em Blocos para Ensino de Programação Paralela com OpenCL

Josué da Silva Gomes Júnior[1], Maelso Bruno Pacheco Nunes Pereira[2], Alisson V. Brito[2],
Jorge Gabriel Gomes de Souza Ramos[3]
[1]*Centro de Ciências Aplicadas e Educação – CCAE*
[2]*Programa de Pós-Graduação em Informática – PPGI*
[2]*Centro de Informática – CI*
[3]*Departamento de Física*
[123]*Universidade Federal da Paraíba – UFPB*
Emails: [josue.gomes, maelso.bruno]@dce.ufpb.br, alisson@ci.ufpb.br, jorgephysics@gmail.com

***Resumo*—Este trabalho apresenta o software educativo Blockly OpenCL. Ele tem por objetivo auxiliar o ensino-aprendizagem do paradigma de programação para arquiteturas paralelas com OpenCL. Nele foi utilizado o paradigma de programação visual, através de um ambiente de desenvolvimento baseado em blocos feito com a API do Google Blockly, que permite ao usuário criar aplicações manipulando blocos e exportá-los para a linguagem OpenCL C++. O Blockly OpenCL também é composto por uma página web contendo uma contextualização da ferramenta, exemplos e materiais de apoio sobre o OpenCL.**

I. Introdução

Utilizados principalmente na indústria de entretenimento, computadores baseados em processadores *multicore* e *many-core* trazem em si também outras arquiteturas distintas da própria CPU, como a GPU *(Graphical Processing Unit)*, promovendo o aumento de desempenho dos sistemas atuais. Esses processadores estão presentes em: *smartphones*, *smart tvs*, *single board computer* e computadores pessoais. Nas últimas décadas estão sendo cada vez mais utilizados em pesquisas científicas [1].

Atualmente, há algumas APIs *(Application Programming Interface)* e *frameworks* que provêm interfaces que facilitam a programação para tais dispositivos, entre elas estão o CUDA, *(Compute Unified Device Architecture)*, o OpenMP *(Open Multi-Processing)* e o OpenCL *(Open Computing Language)*. Diferente dos demais, o OpenCL se popularizou devido proporcionar um padrão para o desenvolvimento em GPGPU (*General Purpose Graphics Processing Unit*), tipicamente adotado por sua portabilidade entre GPUs e CPUs.

Mesmo o OpenCL trazendo algumas facilidades na programação, ainda são necessárias algumas habilidades no desenvolvimento de aplicações paralelas, como programar em baixo nível de abstração, com instruções próximas ao comando de máquina. Assim, é percebido uma rigorosa curva de aprendizagem necessária para lidar com essas APIs [2]. O OpenCL é um assunto que pode ser considerado complexo, isso ocorre devido ao alto nível de conhecimento exigido do desenvolvedor, como conhecer a programação do hospedeiro (*host*), a programação do dispositivo (*device*) e o mecanismo de transferência de dados entre o hospedeiro e o dispositivo [2].

Por outro lado, a linguagem visual vem se mostrando bem promissora no auxílio ao ensino de programação de computadores. Com ela é possível trabalhar com os mais diversos públicos, levando em conta a forma intuitiva de programar, manipulando blocos com funções pré-programadas e convertidas ao modo textual de programação [3]. Essa prática facilita e torna mais ágil o desenvolvimento de aplicações, pois com a junção de alguns blocos, de forma lógica, é possível obter várias linhas de código de um programa procedural, diminuindo a barreira ao aprender ou desenvolver novos programas.

Tendo em vista as vantagens no uso de linguagens visuais, este trabalho objetiva desenvolver um ambiente com uma linguagem visual, baseada em blocos, para apoio à aprendizagem de programação paralela com OpenCL. Esse ambiente tornará fácil e transparente o desenvolvimento em programação paralela com OpenCL. O ambiente foi desenvolvido a partir da API Blockly [4], que permite a criação de novos conjuntos de blocos para a exportação em uma dada linguagem de programação.

Para este trabalho, foi escolhida a API OpenCL como base para a aplicação, pelo fato de ser aberta, livre de *royalties*, com finalidades gerais, funcionar em plataformas heterogêneas e possuir uma comunidade ativa, além de possuir suporte em algumas linguagens como C, C++, Python, Java e Javascript.

O público alvo é desde alunos iniciantes a avançados, que possuam a necessidade de aprender o paradigma de programação paralela, bem como pesquisadores de áreas de conhecimento diferentes da computação e que precisam utilizar programação paralela em suas pesquisas.

Também foi desenvolvida uma página web, para servir como documentação, contendo exemplos e tutoriais. A página segue o conceito de *Web-based learning* (WBL), que inclui algumas vantagens técnicas, como acessibilidade universal, fácil atualização do conteúdo e funções de hiperlink, que permite referenciar para outros recursos. A escolha do formato de informação se deu pelo impacto que essa pode ter na aprendizagem dos conteúdos. O formato hipermídia suporta uma aproximação mais flexível para a instrução, ajudando estudantes a trabalhar com o conteúdo de diferentes perspectivas

[5].

O trabalho está estruturado em 7 seções, divididas da seguinte maneira: a seção II apresenta a Fundamentação Teórica; a seção III aborda os Trabalhos Relacionados; a seção IV apresenta o desenvolvimento da ferramenta; na seção Potencial Pedagógico V na seção VI encontram-se as Considerações finais e Conclusão e na seção VII os Trabalhos Futuros.

II. Fundamentação Teórica

Em razão do alto poder computacional das GPUs, presentes até mesmo nos computadores para fins domésticos, a comunidade científica passou a utilizá-los para computar grande parte dos algoritmos numéricos, auxiliando assim na resolução problemas científicos e da engenharia [6]. Antes esses dispositivos eram utilizados para fins de entretenimento como renderização de vídeos e jogos.

A. Paralelismo e OpenCL

Tendo em vista o rápido aumento na diversificação das plataformas de computação, fez-se necessário um padrão de desenvolvimento para as aplicações, assim surgiu o OpenCL [7]. Mantido atualmente pelo Khronos Group, o OpenCL é uma API e uma linguagem de programação de padrão aberto e livre de *royalties*, compatível com múltiplas plataformas para programação paralela [8]. O objetivo do padrão OpenCL é unificar em um único paradigma e conjunto de ferramentas o desenvolvimento de soluções de computação paralela para dispositivos de naturezas distintas [1]. O OpenCL padroniza o desenvolvimento de programação paralela, pois oferece um conjunto comum de ferramentas para tomar vantagem de qualquer dispositivo com suporte à OpenCL para o processamento de código paralelo, por criar uma interface de programação bem próxima ao hardware [9].

Em contrapartida, A API nativa do OpenCL requer do desenvolvedor uma grande quantidade de "*boilerplate code*"[1]. Outro problema ao começar a programar em OpenCL é a necessidade de familiarizar-se com um novo paradigma de programação [10].

Uma aplicação em OpenCL possui quatro passos que se repetem na maioria dos programas em OpenCL [11], são eles:

1) Levantamento dos dispositivos disponíveis, criação do contexto de execução;
2) Envio de dados para os dispositivos, copiando do *host*, que normalmente é a CPU para os dispositivos.
3) Execução do *Kernel* nos dispositivos e são passados os parâmetros através da chamada da API OpenCL, que é executado de forma assíncrona em relação ao *host*;
4) Leitura dos resultados gerados pela execução nos dispositivos

Além disso, a estrutura do programa em OpenCL se divide em duas partes [12], a primeira parte é chamada de host, programado em C++, onde são definidos os quatros passos descritos anteriormente e é executada na CPU, a outra parte é o *Kernel* programado em OpenCL, que são os programas executados nos *Devices* e compilado em tempo de execução. Essas duas partes podem ser observadas na figura 1.

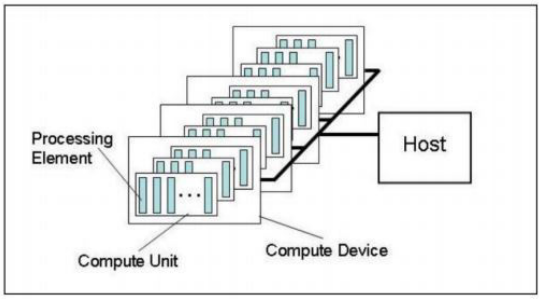

Figura 1. Modelo da Plataforma OpenCL [8]

A execução do OpenCL nos *Devices*, onde se dá o paralelismo, possui alguns detalhes, como:

- **Kernel**: são como funções que são executadas nos dispositivos, os kernels são as únicas funções que podem ser chamadas pelo *host*.
- **Work-Item**: o *work-item* é a menor entidade de execução. Todos os *work-itens* executam uma cópia do *kernel*. Cada *work-item* tem um ID local e um ID global, que são usados para distingui-los.
- **Work-group**: *work-groups* existem para agrupar os *work-itens*, permitindo a comunicação e cooperação entre eles. Assim como os *work-itens* os *work-groups* possuem um ID.

B. Google Blockly

As linguagens de programação visual permitem que o usuário crie programas manipulando elementos gráficos. Como exemplo temos o MIT App Inventor [13] e Scratch [14]. Essas linguagens podem ser a mais forte alternativa existente à prática de programação convencional [15]. O uso de linguagens baseada na conexão de blocos de comandos facilita o aprendizado [16]. Pois, o uso dessas linguagens na aprendizagem de programação garante que os alunos se concentrem muito mais na lógica de programação do que na sintaxe da linguagem [17].

O Blockly é uma interface de desenvolvimento de linguagem visual totalmente *web-based*, na qual os usuários podem arrastar blocos e gerar novos códigos. O Blockly não é uma linguagem de programação, ele apenas converte os blocos em linguagem procedural, tais como Python, Javascript e PHP [4]. O Google Blockly Developer fornece um framework onde é possível criar blocos e exportá-los para a linguagem desejada. Essa abordagem é utilizada para ajudar aprendizes em programação a criar uma conexão entre a linguagem visual e a linguagem de programação tradicional [18].

A natureza visual do Blockly adiciona usabilidade, que faz dele um grande framework para o desenvolvimento de aplicações de aprendizagem para programadores novatos. E

[1]Em Programação de computadores "boilerplate code" refere-se a uma seção de código que deve ser inclusa em vários locais com pouca ou nenhuma alteração, o programador deve escrever muito código para o mínimo de resultado

que mais de 137 milhões de usuários usam o Blockly framework como parte do Code.org [2] e hora do código[3]. Atualmente mais de 5 milhões de estudantes estão em cursos de programação utilizando linguagem de Programação Visual (LPV) [18].

III. Trabalhos Relacionados

O NoooCL [19] é uma API baseada em node que usa o paralelismo do OpenCL em uma linguagem de alto nível, que é o Javascript e vem facilitar a programação em OpenCL para usuários que preferem ou necessitam usar uma aplicação em node ao invés de C ou C++. O NoooCL tem semelhanças com o Blockly OpenCL, pois ele pretende facilitar o desenvolvimento de aplicações paralelas, com a utilização de uma linguagem de alto nível, a diferença é que o Blockly usa blocos para gerar os códigos em C++ e CL e o NoooCL usa bibliotecas para a criação do programa em Javascript.

Foi encontrada apenas a ferramenta OpenBlocks que gera códigos de forma semelhante ao Google Blockly. O OpenBlocks é um ambiente de desenvolvimento baseado em blocos que disponibiliza uma API para desenvolver novos blocos para determinada linguagem [20]. O OpenBlocks é uma ferramenta semelhante ao Blockly, mudando apenas a linguagem a ser desenvolvida os blocos e a aparência, enquanto o Blockly usa Javascript para desenvolver os blocos, o OpenBlocks utiliza Java. O Blockly foi escolhido devido a facilidade de criar os blocos com a ferramenta BlockFactory, a comunidade ativa e por já existir um ambiente que serve como exemplo e se adequa as necessidades do ambiente planejado para o Blockly OpenCL.

Seguindo o mesmo padrão, a ferramenta Block-C é um ambiente de programação visual com blocos que exporta os blocos montados de maneira lógica para a linguagem C. O Block-C foi desenvolvido com OpenBlocks no intuito de facilitar a aprendizagem da linguagem C para novatos. O Block-C tem muitas semelhanças com o Blockly OpenCL, pois tem o intuito de apoiar aprendizagem de uma linguagem de programação através de uma ferramenta de desenvolvimento com blocos.

IV. Desenvolvimento

A solução encontrada para deixar o OpenCL com alto nível de abstração foi utilizar uma linguagem visual. Foi escolhido o Google Blockly por ele apresentar uma documentação bem definida, uma comunidade ativa e também pela aparência dos elementos, além de já disponibilizar uma ferramenta para a construção dos blocos, bem como um ambiente modelo e por ser uma ferramenta *web-based* que não precisará ser feito o *download* ou instalação.

Para o desenvolvimento dos blocos na API Blockly foi necessária uma breve observação da documentação e de exemplos, também foi utilizado o Block Factory para a construção dos blocos, onde é possível selecionar o formato do bloco, as entradas do usuário, os possíveis encaixes e digitar o código em OpenCL C++ que será exportado em código Javascript para ser incorporado ao ambiente.

A. Block Factory

O Block Factory, mostrado na figura 2, é uma ferramenta disponibilizada pela API Google Blockly, que oferece uma interface amigável para a criação de blocos. Com o Block Factory é possível criar todos os tipos de blocos que se façam necessários. É possível definir cores, formas, tipos de encaixes, restrições de conexão, entradas do usuário e textos. A inclusão do bloco no ambiente é feita através de dois códigos, o primeiro é o *Language Code*, que define da aparência do bloco, e o outro é o *Generator Stub*, que define os códigos que serão gerados na linguagem indicada, no nosso caso é o OpenCL.

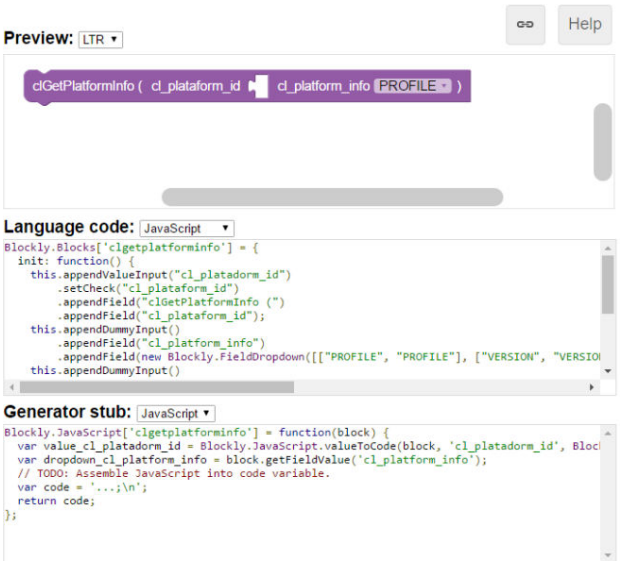

Figura 2. Interface da área de código do Google Block Factory

Devido a uma questão de compatibilidade com os dispositivos, foi escolhido gerar os códigos em OpenCL na versão 1.2, que segue o padrão ISO C99 e com a linguagem C++. Assim, o código gerado a partir da ferramenta será compatível com um maior número de dispositivos presentes no mercado até o momento, já que as versões mais recentes do OpenCL funciona apenas com as plataformas mais recentes.

B. Desenvolvimento do ambiente

A interface de desenvolvimento Blockly OpenCL tem sua estrutura dividida em abas. A aba *blocks*, mostrada na figura 3, corresponde ao ambiente de programação visual, nela encontram-se os blocos que foram arrastados e montados. É possível encontrar botões com opções de controle para o zoom e também o botão de apagar o projeto.

Na segunda aba, figura 4, encontra-se o código OpenCL, gerado a partir dos blocos, que define o *host*. Já na terceira aba ficam os códigos do *Kernel*, que são as funções que

[2]É um projeto que incentiva o ensino de ciência da computação para crianças e adolescentes https://code.org/

[3]É um movimento que visa o ensino de programação através de uma plataforma baseada em programação visual https://hourofcode.com/br/pt

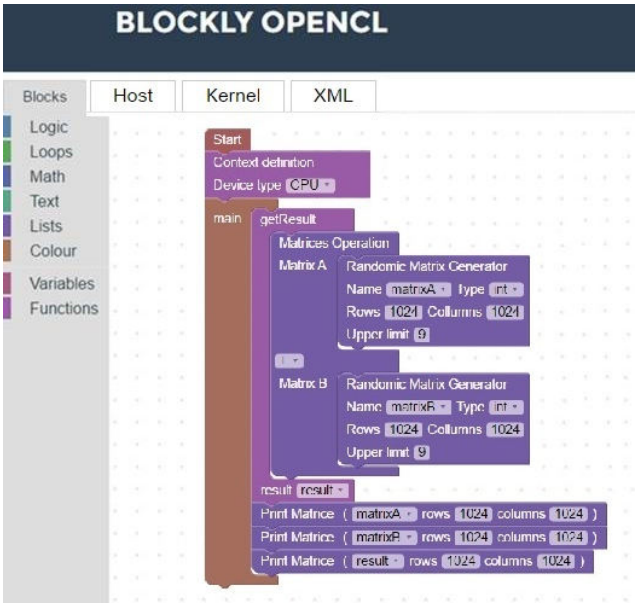

Figura 3. Interface de programação visual

serão executadas nos dispositivos de processamento paralelo. A última aba guarda o código xml, que possibilita salvar o projeto dos blocos.

Blocks | Host | Kernel | XML

```
#define _CRT_SECURE_NO_WARNINGS
#define PROGRAM_FILE "matvec.cl"
#define KERNEL_FUNC "matvec_mult"
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

#ifdef MAC
#include <OpenCL/cl.h>
#else
#include <CL/cl.h>
#endif
int main() {

 /* Host/device data structures */
   cl_platform_id platform;
   cl_device_id device;
   cl_context context;
   cl_command_queue queue;
   cl_int i, err;

/* Program/kernel data structures */
   cl_program program;
   FILE * program_handle;
   char * program_buffer, * program_log;
   size_t program_size, log_size;
   cl_kernel kernel;

/* Data and buffers */
```

Figura 4. Código gerado a partir dos blocos

Percebeu-se ainda a importância de oferecer materiais para que o aluno ou usuário pudesse compreender a plataforma, o paralelismo e as funções do OpenCL. Sendo assim, decidiu-se desenvolver uma página *web* com tutoriais sobre o OpenCL e o ambiente de programação a ser desenvolvido.

C. Desenvolvimento da página Web

Foi desenvolvido uma página web com as informações sobre a ferramenta, exemplos disponíveis comparando a linguagem visual com a linguagem procedural, tutorial da preparação do ambiente, contextualização da programação paralela e do OpenCL e auxiliar as pessoas que queiram aprender a programar com o paradigma de programação paralela. Foi utilizado um template responsivo do Bootstrap[4].

O site serve como base para iniciar estudos sobre o OpenCL, pois além dos tutoriais, a página apresenta um conjunto de informações e sugestões de leitura sobre o tema, ou seja, o site se torna um ponto de partida para as pessoas que queiram aprender a programar em OpenCL. O site tem o propósito de auxiliar com o aprendizado na utilização do ambiente de programação Blockly OpenCL, detalhando passo a passo o uso dos blocos relacionando o código gerado tanto para o *host*, quanto para o kernel, possibilitando ao usuário perceber como funciona o paralelismo no código gerado através dos blocos.

D. Conjunto de blocos que formam um programa com paradigma de programação paralela

Nesta seção será descrito em detalhes um bloco que soma vetores e matrizes de forma paralela, com o intuito de exemplificar de forma didática a conexão entre os blocos, o *Host* e o *Kernel*. Com o arrastar de alguns blocos é possível criar um programa completo, como podemos ver na Figura 5 e descrito detalhadamente nos itens listados a seguir:

- *start* gera no host códigos de definições de arquivos, include da biblioteca OpenCL e cabeçalhos comuns em programas C e C++;
- *Context Definition* seleciona um dispositivo do tipo CPU (em uma lista contendo CPU, GPU, ACCELERATOR, DEFAULT, ALL) para realizar o processamento do *kernel*, cria um contexto e uma fila de comando para o dispositivo;
- o bloco *main* possui a definição do método *"main"* que gera o código de execução no host;
- *RandomicMatrizGenerator*, este bloco é responsável por gerar de forma paralela no dispositivo uma matriz aleatória, com as características passadas como argumentos. Esse bloco foi utilizado duas vezes, uma para cada matriz. Em ambos os casos foi solicitada uma matriz com dimensões 1024x1024, povoada com elementos inteiros, variando de 0 até 9;
- *Matrices Operation*, é responsável por somar MatrizA e MatrixB de forma paralela no dispositivo;
- *getResult*, solicita ao dispositivo que transfira para o *host* a matriz resultante da soma e atribui essa matriz à variável *result*;

[4]O template é disponibilizado para download gratuitamente através do link https://github.com/BlackrockDigital/startbootstrap-freelancer

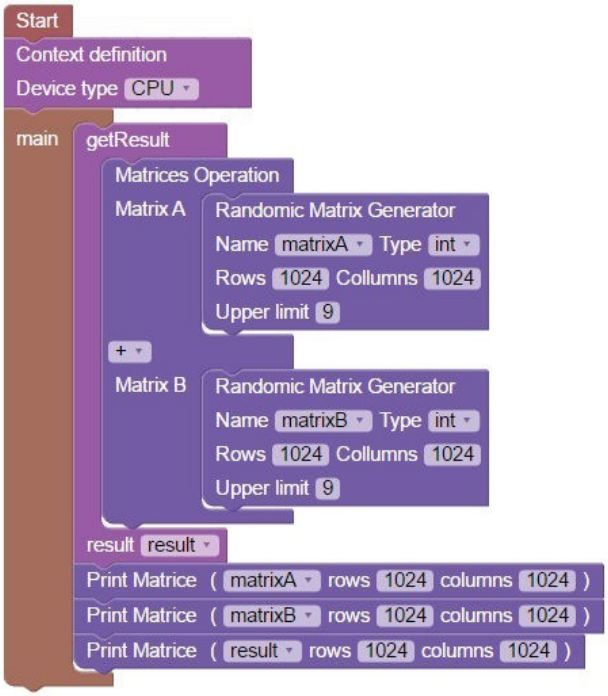

Figura 5. Exemplo dos blocos de soma de matrizes ou vetores

- O bloco *"print matrices"* captura a variável a qual foi alocada as matrizes para ser impressas.

Ao definir essas informações o usuário poderá clicar na aba *Host* para visualizar a definição do *host* e na aba *Kernel* para obter o respectivo códigos do *kernel* OpenCL. Finalmente, deve exportá-los para a IDE e executar o programa de soma ou multiplicação de matrizes ou vetores.

No código de soma de matrizes de forma procedural, a complexidade e o número de passos é muito maior, pois diferente do paralelismo em que todas as somas são feitas em uma única iteração, na linguagem procedural o código irá executar o número de iterações de acordo com o tamanho da matriz.

V. Potencial pedagógico da ferramenta Blockly OpenCL

O ambiente Blockly OpenCL se caracteriza como um software educativo, conforme define Soffa e Alcântara (2012) [21] "o software educativo é desenvolvido com o desígnio de levar o aluno a construir um determinado conhecimento referente a um conteúdo didático. O objetivo de um software educativo é de favorecer os processos de ensino-aprendizagem e sua característica principal é seu caráter didático". Ressalta também que "O emprego destes programas não garantirá por si só a aprendizagem dos alunos, pois os mesmos são instrumentos didáticos de ensino que podem e devem estar a serviço do processo de construção e assimilação do conhecimento dos aprendizes".

Segundo Vieira(1999) [22], o software educativo pode ser classificado em tipos, dentre eles: tutorial, exercícios e práticas, programação, aplicativos, multimídia e Internet, simulação, jogos. O Blockly OpenCL se enquadra em dois deles:

- **Programação:** em que o aluno poderá desenvolver as aplicações e obter o código da aplicação desenvolvida;
- **Tutorial:** no qual a informação é organizada de acordo com uma sequência pedagógica particular, seguindo essa sequência o aprendiz pode escolher a informação que desejar e fazer uma correspondência direta entre cada comando e comportamento do computado

E quanto aos níveis de aprendizagem:

- **Sequencial:** em que o aluno segue as informações que lhe são transmitidas de forma sequencial;
- **Relacional:** em que o aluno pode fazer relações com outros fatos ou outras fontes de informação. Dar-se ênfase na interação do aprendiz com a tecnologia.

Outro fator importante para um software educativo é de se enquadrar em uma teoria de aprendizagem. O Blockly OpenCL adequa-se ao behaviorismo, devido treinar os estudantes a exibir determinado comportamento, usar o reforço positivo para reforçar o comportamento desejado e usar o reforço negativo para reduzir a frequência do comportamento não desejado, além de apresentar o resultado em seções breves, o aluno pode ser testado ao fim de cada seção e apresentar *feedback* imediato [23].

Na educação, o behaviorismo está associado ao trabalho de Skinner, que está focado no comportamento voluntário, deliberado e observável, que ele acreditava ser a maior parte do repertório comportamental de indivíduo [23]. No behaviorismo, aprendizagem é igual a exibir comportamento apropriado. Neste enfoque, a atividade de aprendizagem é planejada de modo a serem ensejadas situações em que o estudante evidencie comportamentos desejados.

Assim como as outras ferramentas, com o paradigma de programação visual citadas neste trabalho, o Blockly OpenCL tem o mesmo potencial pedagógico para apoiar o ensino-aprendizagem de uma linguagem de programação, observando que nesse aspecto, todas elas têm as mesmas características, que possibilitam jovens e adultos a programarem de forma mais simples e ágil, sem a necessidade de conhecer a sintaxe específica de cada linguagem e obter o mesmo resultado. Vale ressaltar que, mesmo possibilitando o desenvolvimento de programas complexos em OpenCL na ferramenta Blockly OpenCL, o usuário necessitará buscar aprender mais sobre a linguagem e o paradigma de programação paralela.

Foge do objetivo desse trabalho que o usuário aprenda a programar em OpenCL apenas fazendo uso da ferramenta. O Blockly OpenCL deve servir como auxiliador para iniciantes nos estudos de OpenCL, agilizar o desenvolvimento de uma aplicação, facilitar testes, possibilitar verificar o código gerado, relacionando-o com o seu respectivo bloco e implementar pequenas mudanças. Também pode ser utilizado por usuários mais experientes, mas não necessariamente para auxiliar o ensino. Neste caso, servirá para desenvolver as aplicações em um menor tempo, tendo em vista que ao arrastar um

bloco é possível produzir, de uma só vez, várias linhas de código. Outra proposta de aplicação da ferramenta é possibilitar pesquisadores de outras áreas de conhecimento que não a ciência da computação, mas que possuam um conhecimento básico em programação, a produzir suas aplicações de maneira paralela, verificando apenas os exemplos e ou tutoriais, além das recomendações de leitura presentes na página web da ferramenta.

VI. CONSIDERAÇÕES FINAIS E CONCLUSÃO

A programação paralela é um paradigma que consiste em computar dados de forma simultânea. Sua utilização pode trazer benefícios relacionados a custos e frequência de processamento. O OpenCL é uma linguagem de programação que faz uso desse paradigma. Ele apresenta um grande potencial no processamento paralelo com volumes elevados de dados com problemas algébricos. Com isso, sua utilização pela academia, apesar de relativamente recente, vêm sendo bem aceita por pesquisadores. Entretanto, devido à complexidade, utiliza-lo ainda apresenta desafios. Sendo assim, acredita-se que a ferramenta apresentada neste trabalho irá contribuir para a solução desse problema.

A avaliação do ambiente de programação Blockly OpenCL ocorreu pela verificação do código gerado pelos blocos. Ao serem colocados de maneira lógica, o código gerado deveria ser similar à sua solução desenvolvida em baixo nível. Ainda não foi possível realizar testes com usuários.

VII. TRABALHOS FUTUROS

Como trabalhos futuros, vê-se a possibilidade de executar a aplicação a partir do ambiente de desenvolvimento do Blockly OpenCL, através de um servidor próprio. Assim, não será necessário fazer o download dos códigos gerados para executá-lo localmente.

Também será realizado testes com alunos e pesquisadores que necessitem utilizar programação paralela, almejando analisar o nível de intuitividade e agilidade ao desenvolver uma aplicação utilizando o Blockly OpenCL. Para assim, saber se a ferramenta realmente contribui no ensino-aprendizagem de programação paralela com OpenCL.

REFERÊNCIAS

[1] C. L. Silveira, L. G. da Silveira Jr, and G. G. H. Cavalheiro, "Programação em opencl: Uma introdução prática," 2010.

[2] M. Scarpino, *OpenCL in Action: How to Accelerate Graphics and Computation*, 2012. [Online]. Available: https://books.google.com.br/books?id=pzuAygAACAAJ

[3] G. da Silveira Júnior, F. Diniz Rossi, P. Lincoln Ramires Izolan, and J. Renan da Silva Almeida, "Análise da ferramenta de programação visual blockly como recurso educacional no ensino de programação," *III Seminário Argentina-Brasil de Tecnologias da Informação e da comunicação*, 2015.

[4] Google Developers, "Blockly instalation," https://developers.google.com/blockly/installation/overview, 2016, acesso em 21 de março, 2016.

[5] T. J. Mitchell, S. Y. Chen, and R. D. Macredie, "Hypermedia learning and prior knowledge: domain expertise vs. system expertise," *Journal of Computer Assisted Learning*, vol. 21, no. 1, pp. 53–64, 2005.

[6] J. Kowalik and T. Puźniakowski, *Using OpenCL: Programming Massively Parallel Computers*, ser. Advances in parallel computing. IOS Press, 2012. [Online]. Available: https://books.google.com.br/books?id=T0sKa4T-sN0C

[7] D. Kirk and W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, ser. Applications of GPU Computing Series. Elsevier Science, 2010. [Online]. Available: https://books.google.com.br/books?id=qW1mncii_6EC

[8] O. W. G. Khronos Group *et al.*, "The opencl specification," *version 1.1*, vol. 1, no. 44, p. 385, 2011.

[9] R. Banger and K. Bhattacharyya, *OpenCL Programming by Example*. Packt Publishing, 2013. [Online]. Available: https://books.google.com.br/books?id=W2lpAgAAQBAJ

[10] D. Demidov, K. Ahnert, K. Rupp, and P. Gottschling, "Programming cuda and opencl: A case study using modern c++ libraries," *SIAM Journal on Scientific Computing*, vol. 35, no. 5, pp. C453–C472, 2013.

[11] A. Tupinambá, "Programação em gpu utilizando opencl," pp. 1–11, 2013. [Online]. Available: http://andrelrt.wdfiles.com/local--files/gpuprogramming:home/Programacao_em_GPU_utilizando_OpenCL.pdf

[12] CodePlex, "Opencl tutorials," http://opencl.codeplex.com/wikipage?title=OpenCL%20Tutorials%20-%201, 2016, acessado junho 11, 2016.

[13] MIT, "Mit app inventor," http://appinventor.mit.edu/explore/, 2016, acesso em 13 de junho de 2016.

[14] MIT Scratch, "Mit app inventor," https://scratch.mit.edu/, 2016, acesso em 13 de junho de 2016.

[15] K. Brock, "Composing accessible code," *Computers and Writing*, 2014. [Online]. Available: http://siteslab.org/cwcon/2014/sites/default/files/public/Brock%20-%20Composing%20Accessible%20Code.pdf

[16] A. B. Finizola, E. H. S. Raposo, M. B. P. N. Pereira, W. S. Gomes, A. L. S. O. de Araújo, and F. V. C. Souza, "O ensino de programação para dispositivos móveis utilizando o mit-app inventor com alunos do ensino médio," in *Anais do Workshop de Informática na Escola*, vol. 20, no. 1, 2014, p. 337.

[17] F. Mélo, R. CUNHA, D. SCOLARO, and J. CAMPOS, "Do scratch ao arduino: Uma proposta para o ensinointrodutório de programação para cursos superiores de tecnologia," in *XXXIX Congresso Brasileiro de Educação em Engenharia, Blumenau, Brasil*, 2011.

[18] C. S. Crawford, M. Andujar, F. Jackson, I. Applyrs, and J. E. Gilbert, "Using a visual programing language to interact with visualizations of electroencephalogram signals," *ASEE-SE Annual Meeting*, 2016. [Online]. Available: http://asee.cs.southern.edu/openconf/modules/request.php?module=oc_program&action=view.php&a=&id=70&type=4

[19] G. Mezős, "Nooocl npm," https://github.com/unbornchikken/NOOOCL, 2016, acesso em 30 de abril, 2016.

[20] R. V. Roque, "Openblocks: an extendable framework for graphical block programming systems," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[21] M. M. Soffa and P. R. d. C. Alcântara, "O uso do software educativo: reflexões da prática docente na sala informatizada," vol. 22, 2008.

[22] F. M. S. Vieira, "Avaliação de software educativo: reflexões para uma análise criteriosa," *http://www. edutecnet. com. br/edmagali2. htm.¿ Acessado em*, vol. 5, no. 11, p. 06, 1999.

[23] K. L. Martins, "Teorias de aprendizagem e avaliação de software educativo," *Monografia) Especialização em Informática Educativa–Universidade Federal do Ceará*, 2002.