Programação visual para introdução ao ensino de programação na Educação Superior: uma análise prática

Elisângela Ribas
IFRS - Alvorada, RS – Brasil – e-mail: elisangela.ribas@alvorada.ifrs.edu.br

Guilherme Dal Bianco
UFFS - Chapecó, SC - Brasil – e-mail: guilherme.dalbianco@uffs.edu.br

Regis Alexandre Lahm
PUCRS - Porto Alegre, RS - Brasil – e-mail: lahm@pucrs.br

Resumo. A aprendizagem de linguagens de programação demanda conhecimentos prévios em lógica e matemática. No entanto, no Brasil, os resultados obtidos por meio do Sistema de Avaliação da Educação Básica (Saeb) demostra haver uma queda no rendimento dos estudantes. A universidade precisa trabalhar com essas questões para atingir seus objetivos, contudo precisa lidar com o alto índice de reprovação e evasão, especialmente em cursos que abordem diretamente saberes lógicos. Nesse contexto, este trabalho apresenta uma metodologia para estimular o conhecimento sobre lógica de programação por meio da substituição de formas tradicionais de ensino por linguagens visuais. A metodologia proposta foi dividida em três etapas, com progressivo aumento do nível de dificuldade. A intenção da divisão foi de não apresentar uma sobrecarga cognitiva aos estudantes, em função da complexidade dos conceitos. Os estudantes alegaram, por meio de avaliação qualitativa, que a utilização de linguagem visual facilitou a introdução de uma linguagem de produção. Além disso, um primeiro resultado demonstrou uma leve melhora nas notas dos estudantes que utilizaram a linguagem visual durante a aprendizagem. Tais resultados, apesar de serem incipientes, demostram que a metodologia é promissora e que novos ajustes podem ser aplicados para sua melhoria.

Palavras-chave: ensino de programação - linguagem visual - estratégias de ensino

Abstract. The programming language learning process requests previous knowledge about math and logic. On the other hand, in Brazil, the results of Evaluation of System Education (Sistema de Avaliação da Educação Básica - Saeb) shows a drop in the learning rate. The University, which is highly impacted by the students' background, should work with high evasion and failed grades. In this context, we propose a new methodology to improve the learning rate of math and logic using a visual language. The methodology has three steps to increase the complexity level progressively. The quantitative experiments, in the real scenario, show that the visual language is easier than the traditional approach to learning the programming language. Moreover, the experimentation shows a smooth improvement in the student's grades. Thus, our initial results show that our methodology is promising and new adjustments can be executed to the continuous improvement.

Keywords: programming learning - visual language - teaching methodologies

1. Introdução

A evasão em cursos superiores é uma realidade não apenas no Brasil. De acordo com Silva *et. al* (2007, p. 3) "verifica-se, em todo o mundo, que a taxa de evasão no primeiro ano de curso é duas a três vezes maior do que a dos anos seguintes." Em cursos ligados à tecnologia, há alguns indícios de que a evasão esteja atrelada a inúmeras reprovações dos estudantes em disciplinas essenciais, como Algoritmos e Lógica de Programação. Barcelos e Silveira (2013), realizaram um mapeamento sobre a temática e constaram que há correlação entre os conhecimentos em Matemática e o sucesso e conclusão do curso por estudantes ligados à área de Computação.

No Brasil, os dados da proficiência em Matemática, de estudantes do Ensino Médio, obtidos por meio do Sistema de Avaliação da Educação Básica - Saeb, demonstram haver um declínio nos resultados ao longo dos últimos dez anos. Em 1995 o resultado foi de 282, passando para 289 em 1996, mas a partir desse ano os números diminuíram progressivamente até chegar em 267 no ano de 2015 (BRASIL, 2015). Embora o raciocínio lógico não seja trabalhado exclusivamente em disciplinas de Matemática, é nessa área que geralmente esses saberes são mais explorados nas escolas. Diante deste cenário, é possível afirmar que cada vez mais os estudantes chegam com menor conhecimento prévio sobre lógica e matemática em cursos superiores.

Feurzeig, Papert e Lawler (2011, p. 6) defendem que ao entrar em contato com a programação de computadores, diversos conhecimentos estavam sendo explorados, entre eles, o de Matemática. “Esta nova abordagem à matemática vai levar a um enorme avanço na capacidade dos alunos para compreender os seus próprios processos de pensamento.” Com a programação, os alunos enfrentam desafios e recebem o *feedback* das ações realizadas, diferente de qualquer outro trabalho abstrato. Ao mesmo tempo que a programação auxilia na compreensão da Matemática, saberes específicos dessa área são fundamentais para a compreensão da lógica de programação.

Utilizar a programação em ambientes educacionais que não estejam voltados a cursos de Computação é uma realidade desde a década de 60, quando Feurzeig, Papert e Lawler no ano de 1968 criaram a linguagem Logo (FEURZEIG, PAPERT, LAWLER, 2011). Logo é uma linguagem de programação, projetada para fornecer uma base conceitual para o ensino de Matemática e as formas lógicas de pensar em termos de ideias e atividades de programação. Tal linguagem deu origem ao Scratch, um ambiente mais interativo, que trabalha com linguagem visual. Essa ferramenta permite que o usuário trabalhe com a programação através da montagem de blocos, fato que facilita a autoaprendizagem e a criatividade para resolução de problemas.

Esse artigo apresenta uma experiência prática de utilização de ferramentas visuais para introdução à lógica de programação em substituição a técnicas de ensino tradicionais, em cursos superiores de Ciência da Computação. O objetivo desse trabalho foi o de compreender os fatores que podem estimular o raciocínio lógico e a aprendizagem de conceitos básicos de programação em cursos superiores. Para a consecução do trabalho, foi proposta a utilização de uma metodologia na qual foram desenvolvidas três etapas para a utilização da linguagem visual. A primeira etapa, com baixo nível de interação em relação à linguagem visual, os estudantes foram convidados a interagir em uma atividade previamente elaborada. Na segunda etapa, com interação média, os estudantes tiveram acesso a comandos básicos de programação (por exemplo, laços de repetição, comandos de decisão e atribuições). Já a terceira etapa, apresentava um nível de interação mais elevado, na qual os estudantes foram desafiados a desenvolver atividades envolvendo lógica e matemática, além disso, os alunos foram desafiados a construir um jogo a partir da ferramenta visual. A ideia foi facilitar o entendimento dos pilares estruturais para a programação, e dessa forma, aprimorar o raciocínio lógico.

A organização do trabalho está realizada em seções. Na seção 2, chamada de Trabalhos relacionados, há uma breve descrição de trabalhos desenvolvidos com temáticas similares. Na seção 3, Desenvolvimento das etapas do estudo, são apresentadas as etapas desenvolvidas para aplicação das propostas. Na seção 4, estão os resultados encontrados e na seção 5 as considerações finais.

2. Fundamentação teórica

Os investimentos para construção de estratégias que busquem contribuir para aprendizagem de programação têm aumentado significativamente nos últimos anos. Souza, Silva e Barbosa (2016, p. 48) realizaram um mapeamento para conhecer os "Problemas e Dificuldades no Ensino e na Aprendizagem de Programação". Como amostra, analisaram 70 trabalhos relacionados à temática, entre os anos de 2010 e 2014. Dentre os principais resultados do estudo, está o fato de que muitos estudantes apresentam dificuldades em compreender conceitos "como ponteiros, recursão e variáveis". Contudo, outros compreendem os conceitos, mas têm dificuldades em colocá-los em prática durante o desenvolvimento de programas. Além desses, os resultados demonstram haver "falta de motivação dos estudantes em realização de atividades de programação". O estudo também apontou alternativas que estão sendo construídas pelos professores para superar tais dificuldades, como "utilização de visualização de programas e algoritmos, *serious games* e o desenvolvimento de ambientes pedagógicos para o ensino e aprendizagem de programação."

Um mapeamento com propósito similar, foi realizado por Silva *et al.* (2015, p. 183) e identificou que, entre as práticas realizadas por professores para o ensino de programação, há uma preferência por "desenvolvimento de ferramentas para o ensino de algoritmos". Além disso, "estudos relacionados indicam que jogos, robótica, ferramentas, metodologias e técnicas" para o ensino de programação contribuem para elevação da motivação dos estudantes e melhores atitudes em relação aos materiais de aprendizagem. Dentre os resultados encontrados no levantamento realizado pelos autores, há indicação de que "a utilização de linguagem de programação visual contribui para facilitar a aprendizagem a medida que os auxiliam com resoluções de problemas".

Diversos são os profissionais que buscam criar estratégias de ensino para tornar a compreensão de conceitos básicos de programação mais acessíveis. Além dos exemplos destacados anteriormente, Koorsse, Cilliers e Calitz (2015), utilizaram as ferramentas de ensino para introdução à computação que envolveu a utilização de *softwares* como RoboMind e Scratch. Os resultados demonstraram que, embora os estudantes perceberam que as ferramentas e estratégias facilitaram a compreensão sobre alguns dos conceitos de programação, não houve indícios de aumento na aprendizagem sobre programação nem aumento da motivação para a programação.

Além desses, França e Amaral (2013, p. 187), descrevem experiências de utilização do Scratch para desenvolvimento do Pensamento Computacional (PC). Entre os conceitos e práticas apresentados, citam que os estudantes puderam “demonstrar competência em sequência, evento, paralelismo, *loop,* condicionais, operadores e dados, sendo paralelismo o conceito menos compreendido e sequência e evento em que apresentaram melhores resultados”. Rodriguez *et al.* (2015, p. 69), descrevem a criação de jogos digitais a partir do Scratch para trabalhar com PC. Dentre os resultados, destacam “a possibilidade de estimular o raciocínio lógico e a resolução de problemas, além de princípios de programação.” Afirmam que “as atividades propostas oportunizaram que os estudantes desenvolvessem habilidades de seleção e organização de informações e se apropriaram dos recursos cognitivos inerentes ao PC.”. Em Souza e Lencastre (2014, p. 265) é relatado que ao trabalhar com conceitos computacionais por meio do Scratch, as turmas melhoraram seus resultados de aprendizagem, além de terem sido expostas à construção e desenvolvimento da competência de resolução de problemas”.

Atualmente, o ensino de programação tem sido estimulado por meio de diversas iniciativas. Fala-se que o conhecimento dessa área pode contribuir para resolução de

problemas em outras instâncias, pois o conhecimento sobre programação e computação estimula o então chamado Pensamento Computacional (PC). Esse termo foi citado pela primeira vez por Wing (2006), que afirmou que PC seria uma das habilidades mais exigidas para o próximo século. Os principais conhecimentos que estimulam o PC são: coleta, análise e representação de dados, abstração, decomposição, depuração, automação, recursividade, iteração, paralelização e simulação (CSTA, 2011).

Wing (2006) afirma que as Ciências da Computação trabalha com muitos princípios de matemática e engenharia. Logo, ao ter contato com a programação, por exemplo, o usuário tende a interagir diretamente com essas áreas, fator que reforçará ou estimulará seu conhecimento lógico-matemático. Por ser uma forma de raciocínio, o PC não tem a pretensão de desenvolver artefatos tecnológicos, como *hardware* e *software*, embora tais ações o estimulem. A programação de computadores contribui para o desenvolvimento de tal pensamento, mas não se restringe a isso. É possível trabalhar com técnicas de programação ou elementos que a integram sem conhecimento específico ou avançado sobre o assunto. A disseminação do Pensamento Computacional oportuniza que os conhecimentos até então específicos das Ciências da Computação possam ser relacionados nos mais diferentes tipos de pesquisa e processos de trabalho.

3. Metodologia proposta

Esta seção descreve a metodologia proposta com objetivo de possibilitar que estudantes com diferentes níveis de conhecimento, sobre lógica de programação, pudessem desenvolver ou aperfeiçoar suas habilidades de raciocínio lógico. As estratégias são baseadas em diferentes níveis de interação, que visam aumentar progressivamente a gama de conceitos. Para possibilitar que os alunos apliquem o conhecimento, foi proposto conjuntos de atividades, cujo objetivo consiste em deslocar um objeto em um mapa até uma (s) posição(ões) específica(s), com o acompanhamento de um professor. As propostas implementadas foram criadas por Zachary Dodds, Professor de Ciências da Computação no Harvey Mudd Collegee e disponibilizadas na plataforma EDX, por meio de um MOOC (Curso Online Aberto e Massivo, do inglês Massive Open Online Course) chamado de *MyCS: Computer Science for Beginners* (DODDS, 2015). Foram divididas em três etapas: interação baixa, interação média e interação alta. Foi utilizado a linguagem visual Scratch, de acordo com Resnick *et al.* (2009), que permite que o aluno encaixe blocos contendo instruções, facilitando o entendimento sobre como solucionar o problema.

Nas etapas de baixa e média interação, foi disponibilizado, para os estudantes, um conjunto reduzido e simplificado de 5 instruções, descritas a seguir: (i) *right*: move o objeto em uma posição para a direita; (ii) *left*: move o objeto em uma posição para a esquerda; (iii) *up*: move o objeto em uma posição para cima; (iv) *down*: move o objeto em uma posição para baixo; e (v) *win*: verifica se ao final do código o objeto atingiu a posição esperada. As instruções são combinadas de forma que o objeto seja deslocado para a posição específica. A seguir será detalhado cada um das etapas.

3.1. Primeira etapa: interação baixa

Essa etapa foi desenvolvida com intuito de oportunizar que estudantes, sem conhecimentos prévios em programação, pudessem construir noções sobre como resolver problemas computacionais, independente do conhecimento sobre lógica. Em outras palavras, a ideia foi possibilitar que todos os alunos tivessem conhecimento sobre o funcionamento das atividades propostas.

Como ponto inicial, os alunos receberam instruções básicas sobre como trabalhar com uma linguagem de programação visual. A ideia foi de oportunizar que, a partir dos comandos disponibilizados, os alunos levassem os objetos até o alvo

específico. A Figura 1 ilustra uma atividade do conjunto de exercícios propostos. No exemplo, o objetivo é deslocar o objeto "peixe" da posição "start" até a posição indicada. Para fazer isso, foi disponibilizada a seguinte sequência de comandos: *right, right, right, win*.

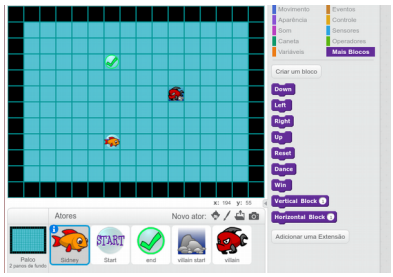


Figura 1: Ilustração da atividade envolvendo um conjunto de instruções reduzidas.

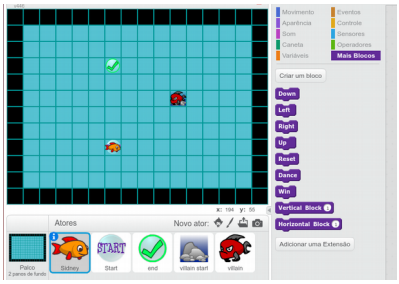


Figura 2: Ilustração da atividade de interação média.

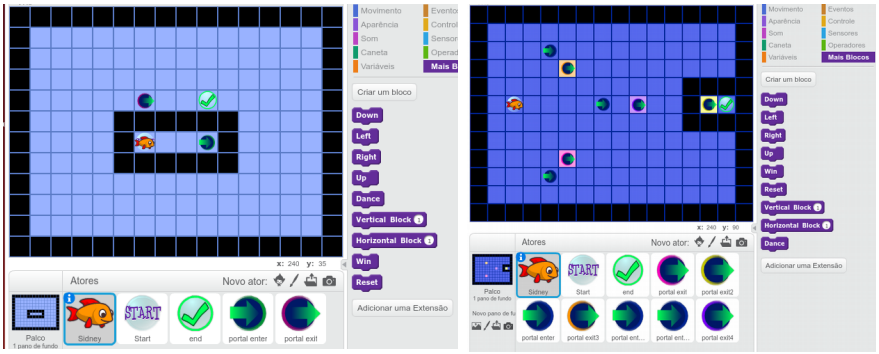
3.2. Segunda etapa: interação média

Nesta etapa, supõem-se que os alunos se familiarizaram com o conceito de instruções e sobre como trabalhar com uma ferramenta de programação visual. O objetivo dessa etapa é avançar com os comandos de programação para que o aluno compreenda estruturas mais complexas, como o controle de repetições e os comandos condicionais. Tais comandos são fundamentais para o desenvolvimento de programas e oferece um primeiro desafio para os alunos iniciantes na programação. Neste momento, os alunos precisaram desenvolver três conjuntos de tarefas com objetivo de, incrementalmente, desenvolver as habilidades lógicas. Cada conjunto foi composto por dez exercícios.

Essa etapa segue a mesma lógica da primeira, na qual o aluno precisou guiar um objeto até um alvo específico. No entanto, além dos comandos básicos, o aluno precisaria gerenciar eventos aleatórios que envolvem a aplicação de comandos condicionais. Para estimular o desenvolvimento dos conceitos, a etapa foi fragmentada em três subetapas. Na primeira, conforme ilustra a Figura 2, o aluno foi desafiado a resolver as atividades utilizando o comando condicional "se", componente fundamental para a aprendizagem de programação. Por exemplo, para definir as instruções é preciso verificar "se" não existe um obstáculo para ser contornado.

Na segunda subetapa, elevou-se o nível de dificuldade dos problemas adicionando o conceito de aleatoriedade. Nesse conjunto de atividades, o objeto "peixe" ao entrar em um "portal" poderia sair em qualquer outro, fato que não existia na tarefa anterior. A Figura 3 apresenta um exemplo dessa atividade, que explora a necessidade do aluno aplicar o conceito de estruturas de repetição por meio do comando "para", já que o objeto pode entrar inúmeras vezes nos portais. Ao compreender a lógica da atividade, o aluno tem condições de construir programas combinando instruções simples (como por exemplo: mover um objeto) até comandos menos intuitivos (como por exemplo: laços de repetição).

No terceiro e último conjunto de tarefas da segunda etapa, a intenção foi de oportunizar que os estudantes combinassem a utilização de comandos de repetição "para" com comandos condicionais "se", conforme ilustra a Figura 4.



Figuras *3 e 4:* Segundo conjunto de atividades na qual o objetivo é aplicar comandos de repetição

Esse conjunto pode ser considerado complexo em relação aos anteriores e buscou consolidar os conhecimentos relativos a esses comandos. Tais tarefas exigem que o aluno seja capaz de controlar estruturas aninhadas de programação, aproximando-se do nível lógico para programar em uma linguagem de produção (por exemplo: JAVA, C, Python, entre outras).

3.3. Terceira etapa: alta interação

Na terceira etapa, os alunos foram desafiados a resolver problemas sem um ambiente controlado, ou seja, nesse momento ele não foram direcionados a utilizar comandos específicos, como "se" e "para" e puderam utilizar qualquer comando disponibilizado pela ferramenta Scratch. A intenção foi de oportunizar que o aluno conseguisse identificar quais instruções deveriam ser combinadas para resolver uma tarefa. Tal fase foi chamada de *alta interação* devido à possibilidade do aluno utilizar uma gama completa de comandos disponibilizados pela ferramenta.

No conjunto proposto, foram utilizados exercícios em que os estudantes são estimulados a encontrar números primos; calcular a área de um triângulo a partir de medidas informadas pelo usuário; simular as operações básicas de uma calculadora, entre outros. Tais atividades oportunizaram que os alunos pudessem relembrar ou exercitar noções de matemática básica aplicadas ao contexto computacional.

Como desafio final, e como instrumento de avaliação, os estudantes precisaram desenvolver um jogo a partir da ferramenta Scratch. O jogo foi proposto para que os estudantes pudessem integrar os diferentes conceitos e comandos, além daqueles apresentados com as tarefas já descritas. A criação de um jogo envolve conhecimentos que extrapolam o que já foi apresentado, além de mobilizar a criatividade e autonomia dos estudantes para irem em busca de soluções para suas propostas.

4. Delineamento experimental

O desenvolvimento desse trabalho surge com objetivo de compreender os fatores que podem estimular o pensamento computacional e a aprendizagem de conceitos básicos de programação em disciplinas de Algoritmos e Programação. Contou com atividades práticas, observação da análise do desempenho e aprendizagem e instrumentos de coleta de dados, com estudantes ingressantes em um curso de Ciências da Computação, no desenvolvimento de uma disciplina de Algoritmos e Programação ao longo um semestre letivo. Além disso, houve comparação com os os resultados obtidos por meio de avaliações parciais e por meio da aprovação em uma mesma disciplina que não utilizou as mesmas metodologias apresentadas.

Inicialmente foi proposto um teste de múltipla escolha com objetivo de analisar os conhecimentos prévios sobre raciocínio lógico e noções básicas de matemática, tais como: área de figuras, diâmetro, regra de três simples e composta e porcentagem. A

intenção da aplicação do pré-teste foi para entender o nível de conhecimento lógico-matemática ingressante no curso. Após, foram aplicadas as três etapas de introdução à lógica de programação, descrita na seção 3. Em seguida, foram utilizadas quatro aulas com 5 horas cada, totalizando 20 horas aula. Os participantes deste estudo foram estudantes de uma turma de 1º semestre do curso de Ciências da Computação de uma instituição de ensino. A coleta de dados ocorreu de Fevereiro de 2016 à Julho de 2016. Participaram da pesquisa cerca de 37 alunos.

Em disciplinas anteriores, as estratégias de ensino para introdução à lógica de programação eram baseadas em Portugol. Trata-se de português estruturado, também conhecido como pseudocódigo e muito utilizado para introduzir conceitos de lógica de programação (MANSO; MARQUES; DIAS, 2010). O Portugol exige que o aluno descreva todas as etapas para construção de um programa, já a linguagem visual oportuniza que o usuário escolha os comandos disponíveis pela ferramenta. Dessa forma, foi realizada uma comparação quantitativa das notas dos alunos na avaliação teórica/prática entre um semestre envolvendo o Portugol e a linguagem visual.

Foram instrumentos para coleta de dados: formulário para avaliação das atividades pelos alunos, com questões dissertativas e de múltipla escolha; avaliação sobre os jogos desenvolvidos e apresentados, primeiro instrumento de avaliação da disciplina; prova aplicada no meio do semestre envolvendo lógica de programação e comparação dos resultados finais entre dois semestres.

5. Resultados e Discussão

Nesta seção, são apresentados os resultados práticos envolvendo a abordagem proposta para a aprendizagem de programação utilizando a linguagem visual Scratch.

Em relação ao instrumento aplicado no formato de teste, com 6 questões que exigiam conhecimento básicos e elementares em lógica e matemática, 32 estudantes responderam de um total de 37 matriculados, com os seguintes resultados: 1 estudante obteve 6 acertos; 4 estudantes tiveram 5 acertos; 13 estudantes obtiveram 3 acertos; 10 estudantes obtiveram 4 acertos. Os resultados podem ser visualizados no Gráfico 1.

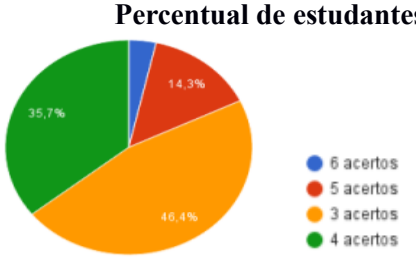


Gráfico 1: Resultado do teste aplicado no primeiro dia de aula

Em outras palavras, o maior número de estudante acertou entre 3 e 4 questões, o que demonstra que os conhecimentos lógico-matemáticos da turma precisariam elevar-se para que fosse possível acompanhar os conceitos mais avançados que seriam trabalhados posteriormente. Por este motivo, foram desenvolvidos exercícios, utilizando Scratch, que envolveram noções de matemática.

Ao final do semestre, os alunos foram convidados a realizar uma avaliação sobre as estratégias utilizadas para aprendizagem no decorrer da disciplina e puderam compararar a linguagem visual com a linguagem de programação C, utilizada no decorrer da disciplina. Do total de matriculados, 25 estudantes responderam ao instrumento apresentado.

Em relação às respostas, quando questionados se já tinham tido contato com alguma linguagem de programação, 68% afirmaram que sim. E, sobre a nota que dariam para o uso do Scratch para introdução de conceitos básicos de programação, em uma

escala de 1 a 5 (ou seja, nota máxima 5), 28% atribuíram nota 5 e 36% atribuíram nota 4. 28% atribuíram nota 3 e 12% atribuíram nota 2. O resultado permite inferir que mesmo não sendo consenso, a maior parte da turma considerou relevante o uso do Scratch. Tais achados são similares aos descritos por Silva *et al.* (2015) e Koorsse, Cilliers e Calitz (2015), que afirmam que os estudantes avaliaram ser mais atrativo e mais fácil de compreender os conhecimentos trabalhados por meio de ferramentas visuais. Feurzeig, Papert e Lawler (2011,) e Wing (2006) afirmam que o conhecimento sobre programação auxilia na compreensão de conceitos matemáticos, mas o desenvolvimento de disciplinas como esta demonstra que o contrário também é verdadeiro. Quanto mais conhecimentos lógico-matemáticos o estudante possuir, mais facilidade na compreensão dos conceitos sobre computação ele terá.

A avaliação consistia de uma pergunta aberta, sendo ela: 1) Comparando o Scratch com a linguagem C, qual sua avaliação? 25 estudantes escreveram comentários, entre eles, a maior parte, 14, informaram que gostaram de ter tido contato com Scratch e que essa auxiliou na construção ou esclarecimentos de conceitos básicos, embora tenham dito que para programar, preferem a linguagem C. Já os outros nove que responderam, disseram que não gostaram de utilizar o Scratch, especialmente em função dos erros que ele apresenta no desenvolvimento de jogos mais elaborados e por isso preferem utilizar a linguagem C.

Os estudantes identificaram problemas na hora de criar programas mais complexos no Scratch e houve quem tivesse perdido todo o jogo em desenvolvimento. Apesar das falhas técnicas que o Scratch apresentou, a avaliação do professor quanto ao uso da ferramenta foi positiva, uma vez que aqueles que nunca tinham tido contato com programação, puderam conhecer alguns conceitos de forma prática e por meio de tentativas e erros. Além disso, a atividade proposta permitiu que os alunos, inicialmente com diferentes níveis de conhecimento, pudessem desenvolver os exercícios.

A avaliação do professor sobre o desenvolvimento das etapas pelos estudantes demonstram que a programação visual tende a contribuir para a aprendizagem básica de programação, uma vez que serve como apoio para introdução de conceitos mais complexos. Essa linguagem visual não pode ser utilizada por muito tempo na disciplina, ou seja, em muitas aulas, pois os alunos tendem a ficar cansados ou encontrar problemas que não se sentem-se motivados a resolver, uma vez que buscam por conhecimentos avançados.

Em relação a comparação entre as notas dos estudantes, o Gráfico 2 ilustra o percentual de aprovados, reprovados e desistentes quando a linguagem visual Scratch foi adotada e quando a linguagem Portugol foi utilizada. Em resumo, a comparação demonstra que houve aumento, mesmo que pequeno, nas notas obtidas por meio das avaliações, uma vez que para o primeiro semestre, quando foi utilizado o Portugol, a média da turma era de 3,6 e quando utilizada linguagem visual a média passou para 4,8. Além disso, a evasão no último semestre de uso do Portugol foi de 20 alunos, já na de programação visual foi de 3 alunos. A princípio, pode-se inferir que a linguagem visual é mais atrativa que o Portugol, mas é necessário novos estudos para conhecer se existem outros motivos que contribuem para evasão e fatores que interferem na aprendizagem dos alunos. Por exemplo, o tempo de dedicação para estudos, o desenvolvimento de atividades extraclasse, os conhecimentos prévios dos estudantes, entre outros.

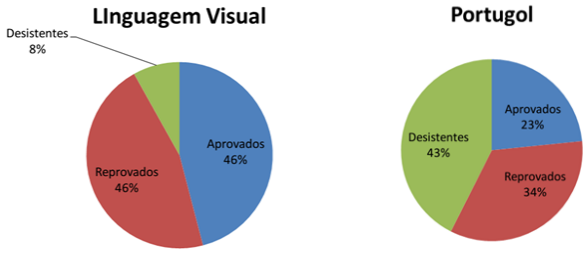


Gráfico 2: Comparação dos resultados finais

Em relação aos conhecimentos prévios sobre lógica e programação que estudantes ingressantes em cursos superiores possuem, é possível inferir, a partir das respostas obtidas no questionário de avaliação dos conhecimentos sobre lógica e matemática, que a maioria dos estudantes apresenta baixo conhecimento nessas duas áreas. Destaca-se que turmas numerosas, em disciplinas que exigem grande envolvimento dos estudantes e que algumas explicações individuais são necessárias, e que muitas vezes o professor não consegue dar a devida atenção a todos, pode contribuir para a desmotivação dos estudantes. Estudos que busquem conhecer as motivações dos estudantes para ingresso e permanência em disciplinas de programação podem ser desenvolvidos a fim de contribuir para o desenvolvimento de novas estratégias de ensino para essa área. Além disso, estudos que contem com apoio a conhecimentos de programação, como oficinas prévias ao desenvolvimento de disciplinas e aulas específicas para tratar de conceitos lógico-matemáticos, também podem indicar resultados positivos para aprendizagem de programação.

6. Considerações finais

Este trabalho teve como objetivo compreender os fatores que podem estimular o raciocínio lógico e a aprendizagem de conceitos básicos de programação em cursos superiores. Para tanto, apresentou uma metodologia para o ensino de programação, que foi desenvolvida em uma turma de Algoritmos e Programação, em um curso de Ciência da Computação. Entre os principais resultados, está o fato de que os estudantes demonstraram maior motivação pela aprendizagem visual, embora entre os resultados de aprendizagem, apresentados a partir da comparação entre as notas de cada semestre letivo, indiquem que houve pequena elevação das notas.

Entre os principais problemas para que a aprendizagem possa avançar, destaca-se o pouco conhecimento que os estudantes apresentam em lógica e matemática ao ingressar nos cursos de áreas consideradas exatas. Tal fato pode estar atrelado aos conhecimentos trabalhados na Educação Básica, mas podem ser de natureza diversa, como o fato de que muitos estudantes não se motivam a estudar tais áreas.

Oferecer atividades extraclasse, como oficinas de lógica e programação podem contribuir para propagar os conhecimentos da área e possivelmente auxiliem os estudantes já matriculados, contudo sabe-se que tais alternativas podem esbarrar em questões administrativas de carga horária docente e de disponibilidade dos estudantes em frequentar esses espaços. Novos estudos que busquem explorar ou aprimorar as estratégias utilizadas podem ser desenvolvidos. Além disso, conhecer o perfil dos estudantes que evadem pode contribuir para a construção de ações preventivas ao longo do curso, especialmente nos primeiros semestres.

Referências

BARCELOS, T. S; SILVEIRA, I. (2013). **Relações entre o pensamento computacional e a matemática através da construção de jogos digitais.** Proceedings of XII SBGames.

Disponível em <http://www.sbgames.org/sbgames2013/proceedings/cd/ _Thiago %20Schumacher%20Barcelos_CD_2013.pdf>. Acesso em 12 de Maio de 2016.

BRASIL. **Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.** Resultados 2015: Prova Brasil. Disponível em < portal.inep.gov.br/web/saeb/resultados-2015>. Acesso em 17 Out. 2016.

CSTA. **Computer Science Teacher Association.** (2011). Disponível em <https://www.csteachers.org/>. Acesso em 10 Maio de 2016.

FEURZEIG, W.; PAPERT, S. A.; LAWLER, B. (2011). **Programming-languages as a conceptual framework for teaching mathematics.** Interactive Learning Environments,19(5):487–501. Disponível em <http://dl.acm.org/citation.cfm?id=965757>. Acesso em 12 Maio de 2016.

FRANÇA, L. R.; AMARAL, H. J. C**. Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do Scratch.** XIX Workshop de Informática na Escola (WIE 2013), pp. 179-187)

KOORSSE, M.; CILLIERS, C.; CALITZ, A. (2015). **Programming assistance tools to support the learning of it programming in south african secondary schools.** Computers & Education, 82:162–178. Disponível em <http://www.sciencedirect.com/ science/article/pii/S0360131514002735>. Acesso em 03 Jun. 2016.

MANSO, A.; MARQUES, C. G.; DIAS, P. (2010). Portugol ide v3. x: **A new environmentto teach and learn computer programming.** In IEEE EDUCON 2010, pages 1007–1010. IEEE. Disponível em <http://ieeexplore.ieee.org/abstract/document/5492469/>. Acesso 15 de maio de 2016.

RESNICK, M., *et al*. (2009). **Scratch**: programming for all. Communications of the ACM, 52(11):60–67. Disponível em <http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf>. Acesso em 15 Maio de 2016.

RODRIGUEZ, C. L. *et al.* **Pensamento Computacional:** transformando ideias em jogos digitais usando o Scratch. XXI Workshop de Informática na Escola. (WIE 2015), pp. 62-70)

SILVA FILHO, R. L. L. S. *et. al.* (2007). **A evasão no ensino superior brasileiro.** Cadernos de pesquisa, 37(132):641–659. Disponível em <http://www.scielo.br/scielo.php ? script=sci_arttext&pid=S0100-15742007000300007>. Acesso em 10 Out. 2016.

SILVA, T. R. *et al* (2015). **Ensino-aprendizagem de programação:** uma revisão sistemática da literatura. RBIE, 23(01):182. Disponível em <www.br-ie.org/pub/index.php/rbie/article/view/2838>. Acesso em 15 de Maio de 2016.

SOUZA, D. M.; BATISTA, M. H. S.; BARBOSA, E. F. (2016). **Problemas e dificuldades no ensino de programação:** Um mapeamento sistemático. Revista Brasileira de Informática na Educação, 24(1):39. Disponível em <http://www.br-ie.org/pub/index.php/rbie/article/view/3317>. Acesso em 10 Out. 2016.

SOUZA, M. R.; LENCASTRE, J. A. **Scratch**: uma opção válida para desenvolver o pensamento computacional e a competência de resolução de problemas. Disponível em <http://repositorium.sdum.uminho.pt/bitstream/1822/29944/1/RuiSousa %26JALencastre_EJML_2014.pdf>. Acesso em 13 Dez. 2016.

WING, J. M. (2006). **Computational thinking.** Communications of the ACM. Disponível em <www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>. Acesso 10 Maio de 2016.

WING, J. M. (2016). **Computational thinking,** 10 years later. Disponível em <https://www.microsoft.com/en-us/research/computational-thinking-10-years-later/>. Acesso em 10 Out. 2016.