Aprendizagem de Algoritmos num Contexto Significativo e Motivador: Um Relato de Experiência

Henrique Monteiro Cristovão

Unidade de Computação e Sistemas
Faculdades Integradas Espírito-santenses (FAESA)
Rua Anselmo Serrat, 199. Ilha de Monte Belo – Vitória – ES – Brasil
hmcristovao@gmail.com

Resumo. *Aprendizagem de algoritmos é um tema de grandes debates, pois existem dificuldades amplamente relatadas e vivenciadas pela comunidade de educadores, e estes, por sua vez, propõe soluções variadas e que algumas vezes divergem em alguns pontos. Este artigo relata uma experiência de sucesso realizada com calouros do Curso de Ciência da Computação da FAESA em Vitória-ES. A experiência teve como cerne a utilização do ambiente de programação Scratch que permitiu a adoção de uma aprendizagem significativa e por conseqüência trouxe para os alunos motivação, que é um dos ingredientes mais difíceis de conseguir nesta práxis pedagógica. Entre os resultados positivos deste trabalho, houve um índice menor de desistências bem como uma melhor apropriação dos objetivos da disciplina. Além disto, espera-se que, com a divulgação deste trabalho, esta prática seja disseminada nos cursos de graduação que tenham o aprendizado de algoritmo em seu currículo e, desta forma, melhorar o nível de aprendizagem dos alunos novatos que irá refletir na formação do profissional.*

1. Introdução

Uma das maiores dificuldades enfrentadas por cursos na área de computação é o ensino de algoritmos para ingressantes que ainda não tiveram iniciação na área de programação. Este problema ainda é amplificado quando se considera que as competências desenvolvidas neste período serão um dos fatores determinantes para o fracasso ou sucesso do aluno ao longo do restante do curso.

Muito educadores se empenham em experimentar novas metodologias ou ferramentas que possam minimizar este problema presente principalmente nos dias atuais cujos alunos vindos do ensino médio cada vez mais trazem posturas inadequadas para este tipo de aprendizagem. Por exemplo, a maioria dos alunos traz consigo o medo de errar, atrapalhando o desenvolvimento de exercícios que precisam de inferências e experimentações para se chegar num resultado satisfatório.

Há discussões sobre qual linguagem iniciar o aprendizado e com qual paradigma, procedural, lógico, orientado a objetos etc, e ainda alguns consensos sobre a prática inicial deste aprendizado apoiado em ferramentas de uso fácil que permitam ao aluno experimentar, descobrir, testar e errar (Hostins, 2007; Barros, 2006; Delgado, 2004; Menezes, 2002). A seção 2 deste artigo irá discutir a aprendizagem de algoritmos numa visão geral.

Em meados de 2007 o MIT Media Lab[1] lançou o ambiente de programação Scratch voltado para crianças a partir de 8 anos de idade. Um ambiente que atende muitas das necessidades de formação de competências do século XXI. A seção 3 deste artigo apresenta detalhes deste software.

Ainda neste ano, no segundo semestre de 2007, foi experimentado o uso desta ferramenta como elemento central no início da disciplina de Algoritmo I do Curso de Ciência da Computação da FAESA-ES. Foram observados resultados positivos. A experiência foi remodelada e reaplicada no primeiro semestre de 2008. Conclusões, observações e detalhes desta prática encontram-se na seção 4.

2. Aprendizagem de Algoritmos

Um dos problemas mais relatados pela literatura e também facilmente vivenciado por qualquer professor que já tenha alguma experiência no ensino de algoritmos é quando este ensino ocorre sob a falta de recursos computacionais onde o aluno possa receber feedback imediato das suas tentativas em resolver os problemas propostos. Ou seja, o aluno necessita de ambientes que possa experimentar, simular, investigar possibilidades e sobretudo errar bastante para que tenha percepção sobre os limites de um algoritmo. A falta de ambientes de teste logo no início da aprendizagem de algoritmo supõe que o estudante já possui um excelente nível de abstração em conjunto com um bom raciocínio hipotético-dedutivo uma vez que ele deve conseguir enxergar os resultados de um programa apenas inferindo sobre o código. É claro que esta abstração deverá ser desenvolvida ao longo de sua trajetória, pois bons programadores são também bons investigadores de erros, e esta formação se faz, por exemplo, com exercícios do tipo "teste de mesa" onde o estudante percorre o programa fazendo o papel do computador.

A atividade de programar possui uma cognição muito rica, pois faz o aluno percorrer o ciclo descrever-executar-refletir-depurar apresentado há bastante tempo por Papert (1994) e muito conhecido e aplicado pela comunidade acadêmica. Este ciclo é muito útil na formação do profissional, porque o faz desenvolver competências associadas às necessidades atuais do mercado de trabalho como as capacidades de planejar, antecipar e simular resultados entre outros.

Na construção de algoritmos o aluno está ativo, ele age, explora, brinca, faz arte, realiza experimentos, antecipa procedimentos, controla suas ações, tem a oportunidade de realizar trocas continuadas entre os colegas, e coordenar uma variedade de conteúdos e de formas lógicas de acordo com a sua capacidade perante o domínio (Fagundes, 1997). Escrever um programa é análogo a escrever um texto, pois o aprendiz realiza tentativas, experimentos até chegar a um consenso e compreender o final do programa ou do texto que está escrito. De fato:

> *"Existe a crença de que só se pode programar o que se compreende perfeitamente. Essa crença ignora a evidência de que a programação, como qualquer outra forma de escrita, é um processo experimental. Programamos, como redigimos, não porque compreendemos, mas para chegar a compreender." (Joseph Weizenbaum) citado por (Vitale, 1991).*

[1] http://www.media.mit.edu/

Este processo experimental gera descobertas, e conseqüentemente a construção de conhecimentos por parte do aprendiz e oportuniza a vivência do ciclo descrever-executar-refletir-depurar.

O domínio da linguagem adotada para se escrever o algoritmo é fundamental para que o autor consiga se expressar de forma correta para solucionar um problema. Tal como enfatiza Delgado (2004) a formalização em linguagem natural facilita o aluno a expor sua solução, a um grupo de colegas. Assim, o grupo é estimulado pelo professor a explorar, questionar e validar a solução apresentada pelo colega. Essa verbalização constrói, sob a intervenção dos componentes da turma, progressiva e interativamente, a formalização considerada satisfatória pelo grupo. O papel do professor nessa etapa é o de minimizar o nível de competitividade e manter o grupo em ação colaborativa e investigativa.

Para exemplificar, segue uma coletânea de ferramentas que auxiliam o processo ensino-aprendizagem de algoritmos, sendo umas antigas e bem conhecidas e outras mais novas:

- **Logo**. Ambiente lúdico de programação onde o usuário controla uma tartaruga na tela do computador. Através do programa ela pode ou não deixar rastro por onde anda e com isto fazer desenhos entre várias outras coisas. É baseado na filosofia do construcionismo defendida por Papert (1994). O Logo possui evoluções em termos de recursos multimidiáticos que foram sintetizados em ambientes como o Micro Words[2] entre outros.
- **Klik & Play**[3]. Permite a criação jogos através de uma linguagem não procedimental e orientada a objetos e a eventos. O usuário escolhe cenários e personagens, que são programados através de uma tabela onde as linhas são as condições, as colunas são os elementos e as células são as ações.
- **VisuAlg**[4]. Muito popular, permite editar, interpretar e executar algoritmos escitos em portugol. Possui visualizador de variáveis que funciona como um depurador e recurso de simulação da "tela" do computador.
- **PROPAT** (Barros, 2006). A abordagem para ensinar programação é apresentar aos estudantes pedaços pequenos de programas ao invés de esperar que eles escrevam programas inteiros a partir do zero. O aprendizado pode ser visto como um processo de reconhecimento de padrões que compara experiências passadas, ou soluções recomendadas por educadores de programação com o programa desenvolvido.
- **AVEP** (Pereira Junior, 2006). É um ambiente que se apóia na tentativa de fazer com que os alunos utilizem o computador a maior parte do tempo, isto inclui a resolução de problemas através de jogos computacionais, interação por meio de um fórum de discussão e o desenvolvimento de programas utilizando uma linguagem de programação, tudo isso via web dando apoio ao estudo assíncrono.
- **Webportugol**. (Hostins, 2007). É uma ferramenta que funciona via web tendo como requisito a máquina virtual Java instalado. Trabalha com portugol, permite a execução passo-a-passo do programa.

[2] Copyright © 1995 Logo Computer System, Inc.
[3] Copyright © 1994 Europress Software Ltda.
[4] http://www.apoioinformatica.inf.br/visualg/objetivos.htm

Nestas ferramentas pode-se observar a forte presença de linguagens de fácil aprendizado. Isto se dá para que a carga cognitiva não fique no aprendizado da sintaxe da linguagem em si, mas na resolução do problema proposto.

3. O Ambiente Scratch

Scratch[5] é um software freeware desenvolvido pelo Lifelong Kindergarten group[6] do Massachusetts Institute of Technology (MIT) Media Lab[7] , lançado em 15 de maio de 2007, destinado a criação de jogos simples, animações, histórias interativas, músicas etc. Está disponível para os sistemas Windows, Mac e Linux[8].

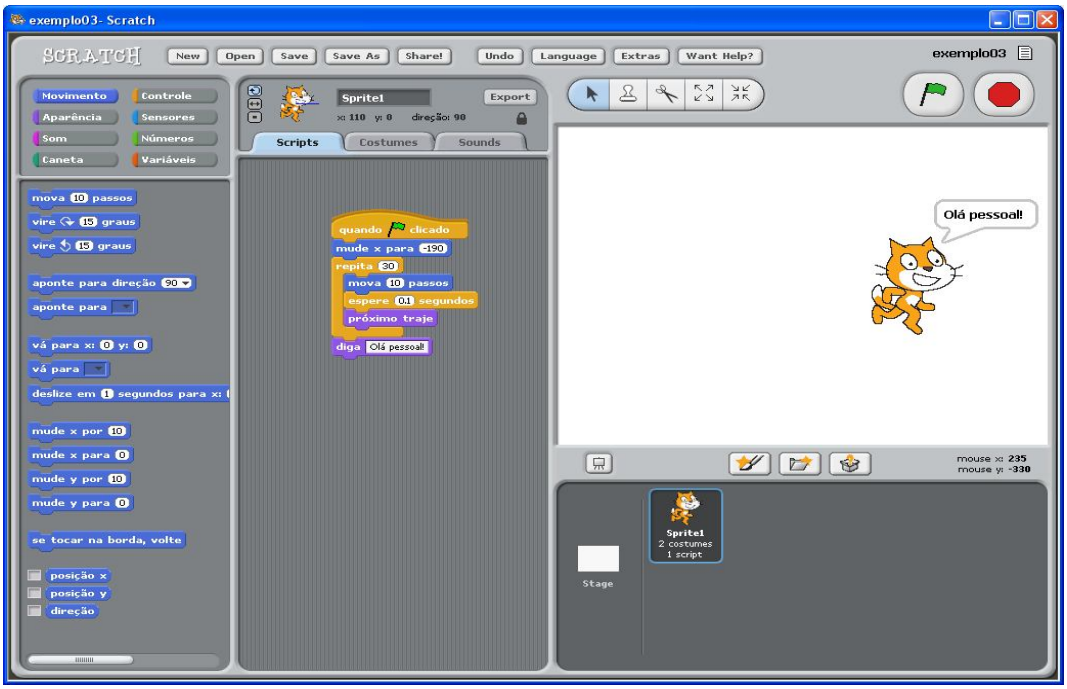

Figura 1. Tela do Scratch com um pequeno programa do tipo "Hello World"

A sua principal característica é possuir um ambiente de programação fácil que não requer conhecimentos prévios de algoritmos ou de programação para o seu manuseio, além de possuir a maior parte da interface traduzida para o português. É indicado para crianças a partir de 8 anos de idade e para pessoas que querem iniciar no mundo da programação de uma forma motivante e prática.

Os projetos desenvolvidos no Scratch podem ser compartilhados num site do próprio MIT de forma muito simples, através de um pequeno cadastro e um simples pressionar de botão. Após o upload, a execução do programa no próprio browser (sem a presença do Scratch) fica disponível para todos, ou então, apenas para os cadastrados, a consulta ao código fonte.

5 http://scratch.mit.edu/

6 http://llk.media.mit.edu/

7 http://www.media.mit.edu/

8 Segundo informações disponíveis no site, a versão oficial para Linux está quase pronta. Apesar disto eles indicam alguns caminhos para rodar neste ambiente.

A interface do Scratch, mostrada na figura 1, é composta de uma área, à esquerda, com os comandos disponíveis que são classificados por cores, uma área central onde o aprendiz constrói o programa sem escrever código, mas escolhendo no menu o comando desejado, e uma área à direita onde vê-se o resultado imediato do programa. Nesta figura observa-se um programa do tipo "Hello World"[9] que já usa uma estrutura de repetição para fazer o "gato" andar pausadamente, e depois exibir num balão o cumprimento.

Segundo Rusk (2007) o Scratch apóia e promove o desenvolvimento de nove competências de aprendizagem importantes para o século XXI. Elas estão detalhadas no wiki da disciplina (http://algoritmo.wikidot.com/scratch).

- Competências no gerenciamento de informações em diversas mídias.
- Competências de comunicação.
- Raciocínio crítico e pensamento sistemático.
- Identificação, formulação e resolução de problemas.
- Criatividade e curiosidade intelectual.
- Competências interpessoais e de colaboração.
- Autonomia.
- Responsabilidade e adaptabilidade.
- Responsabilidade Social.

4. A Experiência Realizada

A experiência foi realizada por dois semestres letivos, 2007-2 e 2008-1, em turmas de calouros do Curso de Ciência da Computação na disciplina de Algoritmo I da Unidade de Computação e Sistemas da FAESA em Vitória/ES. As turmas tinham em média 60 alunos entre ingressantes, transferidos e repetentes. A disciplina possui 68 h/a distribuídas em dois encontros semanais noturnos ao longo de um semestre letivo. O trabalho com o Scratch durou 1 mês e meio, sendo as aulas sempre no laboratório com 2 ou 3 alunos por máquina. No restante do semestre, 2 meses e meio, eles seguiram com a aprendizagem de algoritmos via linguagem Java. Antes os alunos desenvolviam a introdução de algoritmos através de portugol e/ou fluxogramas para depois se iniciarem com a programação na linguagem Java.

Foi criado um wiki para a disciplina, onde os alunos puderam compartilhar seus projetos: http://algoritmo.wikidot.com/scratch. O wiki é um espaço na internet cuja facilidade de edição e o estímulo a cooperação são os elementos que mais conta positivamente. Desta forma o professor fica motivado a manter os dados da página atualizados e ainda a registrar feedbacks de forma rápida para os seus alunos, além de deixar as informações mais organizadas do que, por exemplo, numa lista de discussão (Cristovão, 2007).

Foram preparados cinco roteiros, também disponíveis no wiki, para exemplificar comandos e construções típicas e necessárias e propor problemas para os alunos. Cada roteiro tem um espaçoa no wiki que pode receber contribuições diversas entre dúvidas, descobertas e propostas de soluções.

[9] Tradicionalmente é o primeiro programa que se apresenta no aprendizado de uma nova linguagem de programação: apenas exibe um termo de cumprimento.

Na figura 2 encontra-se um programa exemplo, disponível no primeiro roteiro, que pergunta ao usuário a sua idade e em seguida exibe se ele é maior ou menor de idade.

Figura 2. Exemplo de programa com estrutura de repetição e condicional

Em outro exemplo, figura 3, vários quadrados são desenhados através de um recurso, muito parecido com a linguagem Logo, que faz o personagem, neste caso a "formiga", deixar um rastro por onde passa.

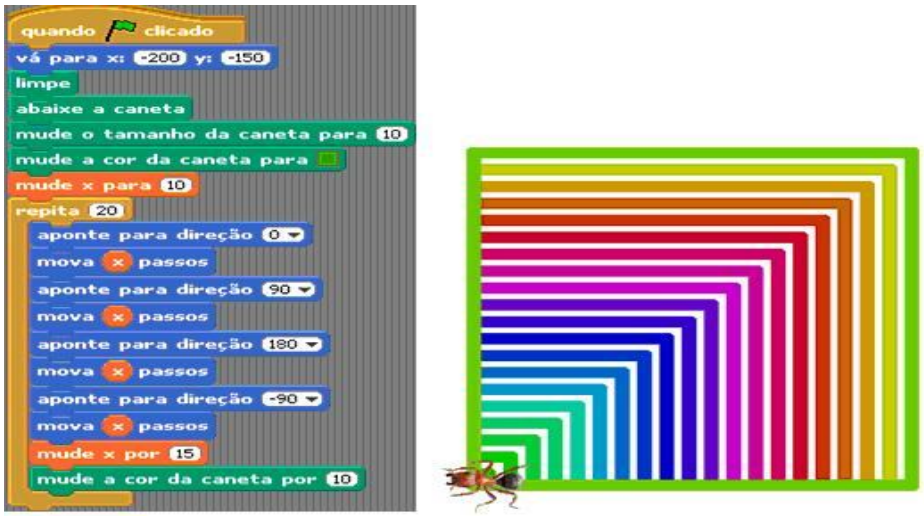

Figura 3. Exemplo de programa com estrutura de repetição e variável acumuladora

A figura 4 mostra um programa com duas estruturas de repetição aninhadas: o comando repita mais interno faz uma estrela com 20 pontas, e o repita mais externo determina que serão 10 estrelas ao todo. O resultado do programa por ser visto nesta mesma figura.

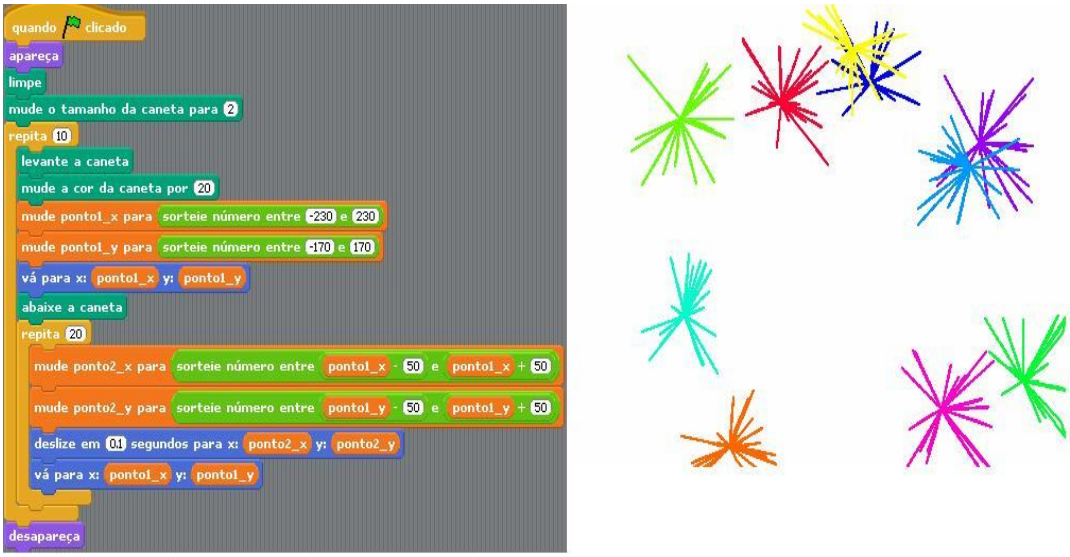

Figura 4. Exemplo de programa com estrutura de repetição aninhada

A avaliação da aprendizagem foi baseada no domínio das estruturas algorítmicas que cada aluno apresentou. Para isso a escolha do projeto, a ser construído e avaliado, foi livre permitindo assim que o aluno se responsabilizasse pela criação e desenvolvimento do tema. A conseqüência disto foi que a relação entre o aluno com o projeto ficou bem mais próxima. Cada aluno apresentou o seu projeto para a turma num tempo limitado. Foi estimulada a construção cooperativa entre os vários autores, apesar da responsabilidade do projeto pertencer a somente um aluno. Desta forma evitou-se que um determinado aluno pudesse participar de forma passiva e com isto não vivenciar o processo de criação do projeto de maneira responsável e ativa. Apesar do trabalho maior para o professor, todos puderam fazer o papel de autor da idéia do projeto e de apresentador. A avaliação de cada autor de projeto ocorreu no momento de sua apresentação através de perguntas que sondaram sua segurança e verificaram as estruturas algorítmicas usadas:

- Variáveis simples e variáveis acumuladoras
- Estrutura condicional: **se...**
- Estrutura condicional completa: **se...senão...**
- Estrutura de repetição: **sempre, repita....**
- Estrutura de repetição **repita ...** com variável contadora
- Estrutura de repetição com condicional: **repita até ..., para sempre se ...**
- Estruturas de repetição aninhadas
- Estrutura de repetição aninhada com estrutura condicional ou vice-versa

Além disto, foram medidos também outros critérios que puderam valorizar a criatividade, originalidade, clareza e organização da apresentação, extensão do projeto e elementos extras como: a jogabilidade, presença de fases e níveis se o projeto fosse um jogo, nível de interatividade, recursos extras do software etc.

Vários alunos que cursaram esta disciplina com outro método/ferramenta ficando reprovados, relataram que na experiência com o Scratch conseguiram se adaptar melhor, se apropriar melhor do conhecimento e sentiram-se mais seguros em desenvolver

programas. Seguem alguns pontos positivos observados nesta experiência no contexto do aprendizado de algoritmos na graduação:

- **Motivação**: maior motivação devido a presença de significado no manuseio do software, pois os resultados são imediatos e fazem com que os alunos criem rapidamente situações de seu interesse. Assim, eles demonstraram muito mais ávidos em fazer as atividades propostas.
- **Qualidade na aprendizagem**: a apropriação das estruturas algorítmicas foi nitidamente maior e num tempo menor, se comparado ao desenvolvimento com portugol ou fluxogramas.
- **Menor impacto na transição para Java**: o início do aprendizado na linguagem Java foi mais suave, pois muitas estruturas algorítmicas já tinham sido efetivamente dominadas.
- **Rapidez no aprendizado de estruturas mais complexas:** logo na primeira aula os alunos já se depararam, naturalmente, com estruturas de repetição e condicionais. Mais algum tempo eles já estavam aninhando repetição com condicionais e/ou com outras de repetição.
- **Feedback imediato**: o aluno deve receber feedback imediato sobre as suas inferências e construções. Através de pesquisa, Menezes (2002) detectou grande dificuldade quanto ao acompanhamento individualizado do aluno no que tange a solução de problemas. Quanto a comparação de soluções, devido ao comprometimento da carga horária de sua disciplina, o professor na maioria das vezes considera apenas as soluções de alguns alunos da classe, conduzindo a turma as devidas conclusões. Com a prontidão e facilidade do Scratch em aceitar modificações do programa enquanto o executa e contando com o envolvimento da turma no laboratório o aluno tem possibilidade de receber resposta mais rápida de seus pares sem ter que depender exclusivamente do professor.
- **Modularização de programas:** normalmente a criação de procedimentos (procedimentos, funções ou métodos) é trabalhada no segundo semestre do curso, em Algoritmo II. Com o Scratch o aluno naturalmente sente a necessidade e já cria módulos de código sem mesmo uma explicação prévia do professor.
- **Facilidade com programação paralela:** num curso de computação este tipo de programação, normalmente, só é trabalhada após vários semestres letivos, mas no Scratch é comum que um projeto possua trechos de programação paralela pois eles se iniciam ao mesmo tempo a partir do acontecimento de um evento. Apesar disto, nesta experiência tentou-se minimizar o uso deste tipo de programação porque senão seria mais difícil a passagem para a linguagem Java.
- **Valorização dos veteranos**: os alunos que já possuem experiência em programação têm grande oportunidade de mostrar sua bagagem já que eles poderão apresentar projetos maiores. Isto trás duas vantagens: motiva a participação destes alunos valorizando o seu conhecimento prévio, e imerge os alunos iniciantes em experiências mais complexas e elaboradas feitas pelos seus pares.
- **Cooperação**: com a utilização do wiki, onde o aluno publicava o projeto de sua autoria, outros colegas podiam aprender com estes programas. Durante a aula era

estimulada a circulação dos alunos nas bancadas do laboratório favorecendo as trocas de idéias, experiências, erros e descobertas. Como o Scratch possui um resultado que pode variar de uma simples seqüência animada até um jogo completo os estudantes tinham curiosidade em investigar o que estava acontecendo em outros grupos.

- **Mais tempo programando**: a motivação era visível entre os alunos e isto trouxe um dos elementos mais importantes para um aprendiz de algoritmo: mais tempo dedicado a tarefa de programar trazendo mais vivência e amadurecimento.
- **Menor desistência**: a disciplina de algoritmos é uma das que mais provoca desistência num curso de computação. Após a adoção do Scratch houve uma quantidade menor de desistentes.
- **Valorização de todos o alunos:** mesmo aqueles alunos que tem mais dificuldades na programação tem seu projeto valorizado pois ele pode compensar em outras vertentes como a criatividade no arranjo dos elementos e até elaborando um enredo mais extenso.

Exemplo de um problema simples passado como exercício para os alunos:

> *"Desenvolva aquele velho e conhecido jogo de adivinhação: o computador sorteia um número de 1 a 100 e pede que o usuário tente adivinhá-lo. A cada resposta do usuário o computador dá dicas alertando que o número é maior ou menor, ou então anuncia que ele acertou. Ao término, ainda mostra com quantas tentativas ele conseguiu adivinhar."*

Neste caso o estudante já trabalha com estrutura condicional aninhada numa estrutura de repetição. Outro exemplo de exercício pode ser observado na figura 5, onde o aprendiz descobre a importância do laço com variáveis contadoras para fazer a espiral e o laço aninhado para fazer várias espirais. A relação completa de exercícios usados pode ser obtida no wiki da disciplina.

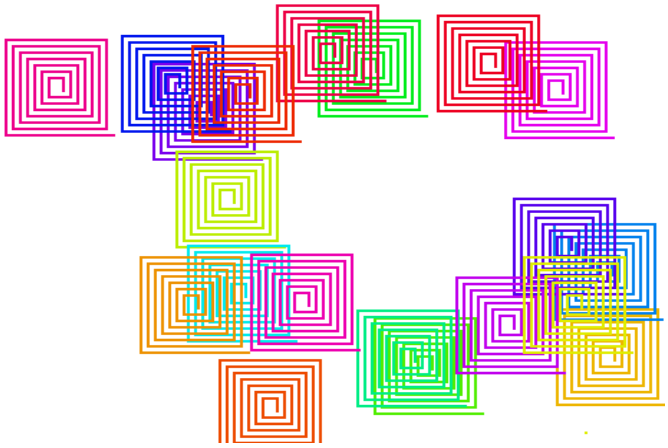

Figura 5. Exemplo de exercício

Apesar de tudo que foi apresentado, observou-se que o Scratch possui algumas limitações e, desta forma, a sua utilização, em cursos de graduação na área de Computação e Engenharia, fica restrita ao início do curso ou mesmo no início da disciplina de introdução a algoritmos. Limitações observadas:

- Dificuldade na entrada de dados via teclado, principalmente strings.
- Falta de estruturas de armazenamento homogêneas: vetores e arrays.
- Falta de ponteiros, inviabilizando o tratamento mais minucioso da memória.
- Falta de procedimentos com argumentos.
- Falta de comentários no programa.

5. Conclusões

O ensino de algoritmo tem constantemente recebido apoio com o surgimento de ambientes que permitem ao aluno aumentar o seu grau de aprendizagem tanto dentro da sala de aula quanto fora, permitindo o desenvolvimento da autonomia do aluno.

Um dos fatos que mais chamou a atenção nesta experiência foi a observação de que os alunos, logo na primeira aula da disciplina, que acontece no laboratório, já saíram sabendo na prática o significado de um algoritmo computacional e também dominando diversas estruturas de programação que normalmente iria aprender após muitas aulas. Desta forma o aluno fica motivado a continuar seu aprendizado e, por conseqüência, a sua autonomia é desenvolvida. Ele obtém resultados rápidos a partir de pouca programação, mas em outro momento descobre que um bom programa requer bastante planejamento e árduo trabalho.

Outro fator a ser destacado é o respeito ao conhecimento do aluno que já vivenciou em algum momento a construção de algoritmos, quer seja no ensino técnico ou de forma autodidata, mas de qualquer forma este aluno fica motivado a usar o conhecimento que já possui para alcançar objetivos maiores. E isto ocorre sem que os alunos iniciantes fiquem perdidos no processo, pois, devido as características do ambiente, eles também têm condições de construir produtos interessantes.

Uma vertente importante é a aplicação deste ambiente no ensino médio, pois conseguiria preencher uma grande lacuna de competências existente nos alunos que chegam ao nível superior e querem cursar uma graduação na área de computação. O trabalho com o ensino médio mostra-se importante também para aqueles alunos que seguirão outras áreas de atuação. Tal como afirma Resnick (2007) e mostrado na seção 3.1 deste trabalho, aprender a programar traz benefícios a todos: permite aos alunos expressarem-se de forma mais completa e criativa, ajuda-os a desenvolver o pensamento/raciocínio lógico e ajuda-os a compreender o funcionamento das novas tecnologias que encontrarão por todo o lado na sua vida profissional.

6. Referências Bibliográficas

BARROS, Leidiana Nunes de; DELGADO, Karina Valdivia. Aprendizado de programação. In.: Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Campo Grande, 2006.

CRISTOVÃO, Henrique Monteiro. Wiki como estratégia de aprendizagem no ensino a distância ou presencial. In: WEI - VII ERI (Escola Regional de Informática) - SBC, 2007, Vitória-ES.

DELGADO, C.; XEXEO, J. A. M.; SOUZA, I. F., CAMPOS, M., RAPKIEWICZ, C. E. Uma abordagem pedagógica para a iniciação ao estudo de algoritmos. In.: Workshop

de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Salvador, 2004.

FAGUNDES, Léa da Cruz (Org.). Informática na escola: pesquisas e experiências. Brasília : MEC/UNESCO, 1994.

HOSTINS, Higor; RAABE, André. Auxiliando a aprendizagem de algoritmos com a ferramenta Webportugol. In.: Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, 2007.

MENEZES, Crediné Silva de; NOBRE, Isaura Alcina Martins. Um ambiente cooperativo para apoio a cursos de introdução a programação. In.: Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Florianópolis, 2002.

PAPERT, Seymour. A máquina das crianças : repensando a escola na era da informática. Porto Alegre : Artes Médicas, 1994.

PEREIRA JUNIOR, J. C. R.; RAPKIEWICZ, C. E.; XEXEO, J. A. M.; DELGADO, C. AVEP – um ambiente de apoio ao ensino de algoritmos e programação. In.: Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Campo Grande, 2006.

RESNICK, Mitchel. Rethinking learning in the digital age. The Media Laboratory. Massachusetts Institute of Technology. 2007. Disponível em <http://www.media.mit.edu/~mres/papers/wef.pdf>

RUSK, Natalie; RESNICK, Mitchel; MALONEY, John. 21st century learning skills. Lifelong Kindergarten Group. MIT Media Laboratory. 2007. Disponível em <http://llk.media.mit.edu/projects/scratch/papers/Scratch-21stCenturySkills.pdf>

VITALE, Bruno. Computador na escola: um brinquedo a mais. Revista Ciência Hoje, v. 13, n. 77, p. 19-25, nov. 1991.