

**iVProg : um sistema visual
para ensino/aprendizagem
de programação via Web**

Reginaldo Rideaki Kamiya

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS
PROGRAMA CIÊNCIA DA COMPUTAÇÃO

Orientador: Prof. Dr. Leônidas de Oliveira Brandão

São Paulo, dezembro de 2010

iVProg : um sistema visual para ensino/aprendizagem de programação via Web

Este exemplar corresponde à dissertação de mestrado de Reginaldo Rideaki Kamiya

Banca Examinadora:

- Prof. Dr. Leônidas de Oliveira Brandão (orientador) - IME-USP.
- Prof. Dr. Marco Aurélio Gerosa - IME-USP.
- Profa. Dra. Maria Alice Grigas Varella Ferreira - POLI-USP.

Agradecimentos

Meus agradecimento ao professor Leônidas de Oliveira Brandão, pela paciência e dedicação durante o mestrado.

Aos meus pais, Flávio e Kinuko, e à minha irmã Sati, por todo o apoio, dedicação e confiança que depositaram em mim.

À minha namorada Tarsilla, pelo apoio, paciência, incentivo e compreensão.

Aos meus amigos, Everaldo, José, Patrícia e Eliane pela excelente companhia e incentivo que me proporcionam desde o início do mestrado.

A todos que direta ou indiretamente me ajudaram a chegar até aqui.

À Carnegie Mellon University que disponibilizou o código fonte do sistema Alice.

Resumo

A literatura aponta uma grande dificuldade de aprendizado em disciplinas de introdução à programação em cursos de ciências exatas. Pode-se encontrar vários métodos propostos para atacar esse problema, entretanto poucas são as iniciativas focadas em sistemas Web ou em sistemas visuais. Nesta dissertação apresentamos um novo sistema Web, cuja programação é feita de modo visual. Também apresentamos experimentos que indicam que este modelo visual reduz significativamente as dificuldades dos alunos.

Palavras-chave: ensino de programação, algoritmos, sistemas Web.

Abstract

The literature points out that learning algorithm and programming is a difficult task. Several methods were proposed to overcome this problem, but just a few initiatives focused on Web systems or on systems with a visual model of programming. This paper presents a new Web system to teach/learn algorithms, whose programming is performed visually. It also presents some positive results of this new free software during experiments conducted with undergraduate students.

Keywords: programming teaching, algorithms, web systems.

Sumário

Lista de Abreviaturas	ix
Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
1.1 Programação visual	2
1.2 Aprendizagem via Web	4
1.3 Delimitação do problema	5
1.4 Justificativas	6
1.4.1 Uso via Web	6
1.4.2 Uso da programação visual	6
1.5 Contribuições	6
1.6 Objetivos	7
1.7 Organização do Trabalho	7
2 Trabalhos relacionados	9
2.1 Novos métodos	9
2.2 Novos sistemas	11
2.2.1 Sistemas de ensino <i>off-line</i>	12
2.2.2 Sistemas de ensino <i>on-line</i>	22
3 O sistema iVProg	27
3.1 Descrição geral do iVProg	28
3.1.1 Aplicativo/applet	28
3.1.2 Modelo de programação (visual) do iVProg	29
3.1.3 Internacionalização	31
3.2 Enriquecendo páginas Web com o iVProg	32

3.2.1	Gerando uma página Web com o iVProg	33
3.2.2	iVProg como Widget	34
3.3	iVProg como iMA	35
4	Experimentos	39
4.1	Exercícios observados	39
4.2	Uso em disciplina regular	42
4.2.1	Comparativo de desempenho com turmas anteriores	50
4.3	Conclusões do capítulo	52
5	Conclusões	53
5.1	Considerações finais	53
5.2	Contribuições	54
5.3	Trabalhos Futuros	55
A	Manual do iVProg	57
A.1	Menu do sistema	57
A.2	Botões de execução e edição	59
A.3	Aba de variáveis	60
A.4	Aba de métodos	60
A.5	Implementação do método	61
A.6	Aba de funções	62
B	Relatos obtidos nos questionários	65
	Referências Bibliográficas	71

Lista de Abreviaturas

AMBAP	Ambiente de Aprendizado de Programação
IDE	(<i>Integrated Development Environment</i>) Ambiente integrado para desenvolvimento de software
iMA	Módulo de Aprendizagem Interativa na Internet
SAW	Sistema de Aprendizagem pela Web
SGC	Sistema Gerenciador de Cursos

Lista de Figuras

1.1	Exemplo de algoritmo utilizando a <i>programação visual</i>	3
2.1	Tela principal do Portugal.	13
2.2	Exemplo de algoritmo no BlueJ.	14
2.3	Exemplo de algoritmo no BlueJ.	15
2.4	Tela do Jeliot.	16
2.5	Tela principal do VisuAlg.	17
2.6	Tela do AMBAP.	18
2.7	Exemplo de criação no Toontalk.	19
2.8	Tela principal do Greenfoot.	20
2.9	Exemplo de edição de algoritmo no Greenfoot.	21
2.10	Exemplo de criação de um jogo no GameMaker	22
2.11	Tela do WebPortugol.	23
2.12	Tela principal do Scratch.	24
2.13	Tela principal do Alice (original).	25
3.1	Contexto destacado na <i>programação visual</i> do iVProg.	30
3.2	Código com os parâmetros necessários para a chamada do iVProg em uma página Web.	32
3.3	Funcionalidade criação de páginas Web com o iVProg.	33
3.4	Exemplo de programa/algoritmo no iVProg.	34
3.5	Página Web com o programa gerado na figura 3.4.	35
3.6	Widget do iVProg.	36
3.7	iVProg como iMA.	37
4.1	Frequência de uso do computador.	43
4.2	Opinião dos alunos com relação ao aspecto visual do iVProg.	44
4.3	Opinião dos alunos em relação ao primeiro contato com o iVProg.	45
4.4	Opinião dos alunos com relação à resolução de exercícios no iVProg.	46

A.1 Tela do iVProg.	58
A.2 Menu arquivo	58
A.3 Menu exercício	59
A.4 Menu ajuda	59
A.5 Botoes e o icone 'Lixo'.	60
A.6 Aba de métodos.	61
A.7 Aba do método no iVProg	61
A.8 Exemplo de implementação de método.	62
A.9 Destaque das cores em locais em que o componente selecionado pode ser utilizado.	62
A.10 Funções do iVProg.	63

Lista de Tabelas

2.1	Análise dos Sistemas para Ensino de Programação.	11
4.1	Comparativo entre iVProg e a linguagem C nos critérios observados.	41
4.2	Comparativo das 2 turmas	51
4.3	Comparativo dos exercícios feitos em T1.	52

Capítulo 1

Introdução

A disciplina de Introdução à Computação aborda os princípios da lógica de programação com o objetivo de desenvolver a capacidade dos alunos para analisar e resolver problemas na forma de algoritmos. De acordo com BORGES (2000), esta disciplina costuma ter altos índices de evasão e reprovação, sendo um dos gargalos existentes nos cursos de exatas, dificultando ou até mesmo impedindo a continuidade dos estudantes nos cursos.

Dentre os vários motivos citados como causa deste problema, BORGES (2000) e BAEZA (1995) citam duas dificuldades dos alunos no aprendizado de programação: (d1) dificuldade com o ambiente para programação; e (d2) dificuldade com a sintaxe das linguagens tipicamente utilizadas (como Java ou C). Já TOBAR (2001) aponta a necessidade dos alunos resolverem eles próprios vários exercícios/problemas.

Paralelamente, tanto no Brasil quanto em outras partes no mundo, ocorre uma grande expansão do uso da internet no ensino e, conseqüentemente, a demanda por pesquisas nesta área têm aumentado consideravelmente (LITTO & FILATRO, 2004). Através dos avanços da internet e das ferramentas de suporte à educação a distância, tornou-se possível difundir o conhecimento de forma extremamente rápida e atender as demandas por cursos com flexibilidade de horário e local. Neste contexto, os ambientes virtuais se transformam em salas de aula, onde alunos e professores se comunicam e interagem através de recursos como *chats*, fóruns de discussão e *e-mails*.

Neste contexto, o objetivo deste trabalho é apresentar e discutir um novo sistema que visa reduzir algumas dificuldades inerentes aos ambientes tradicionalmente utilizados para se introduzir programação e com isso possibilitar que o aluno concentre-se no que realmente interessa, que é aprender a construir algoritmos. O sistema proposto, denominado por iVProg

- Sistema para programação visual, funciona plenamente em navegadores Web e pode ser integrado a *Sistemas Gerenciadores de Cursos (SGC)*¹.

Este projeto utilizou como base o sistema *Alice*², por este atender a 3 importantes requisitos: ter seu *código livre* (RAYMOND, 2000), ser implementado em *Java* (o que possibilitou portá-lo para a Web) e dispor de um mecanismo visual para programação.

Entretanto foi necessário um grande esforço para viabilizar uma versão do *Alice* que pudesse ser utilizada na Web na forma de *applet*³, pois o sistema estava na forma de aplicativo e com um tamanho (em *bytes*) que inviabilizaria sua carga como *applet*: atualmente o *Alice* tem 140Mb. A atual versão do novo sistema, *iVProg - programação visual na internet*, já está disponível como *applet* e tem apenas 1,4Mb de tamanho.

Este sistema está disponível no endereço <http://www.matematica.br/ivprog> e deverá ser brevemente distribuído na forma de *software livre* (gratuito e com código fonte disponível).

1.1 Programação visual

A *programação visual* permite que o aluno implemente algoritmos baseados em componentes gráficos, assim, para construir seu programa o aluno usa o *mouse* para selecionar os comandos e para definir o local onde estes serão utilizados, usando seus representantes gráficos, sem a necessidade de digitar comandos.

A figura 1.1 exemplifica um algoritmo utilizando esse modelo no *Alice* e no *iVProg*, onde pode-se ver que cada comando é representado por um bloco com cores distintas, além de automaticamente “identá-los”. Deste modo fica mais fácil o aluno identificar o escopo do comando.

A principal vantagem deste modelo é reduzir o esforço cognitivo do aluno, por praticamente eliminar a necessidade de “decorar” comandos e os detalhes de sintaxe, como na linguagem *C*, a diferenciação das instruções de atribuição ($a = b$) e de comparação ($a == b$). Como será mostrado no capítulo 4, esse modelo reduz bastante os enganos de sintaxe

¹ Na literatura é possível encontrar várias denominação para esses sistemas, com diferenças semânticas, como *Ambiente Virtual de Aprendizagem*, *Learning/Content Management System*. Neste texto usaremos o termo *SGC* para designar qualquer sistema que realize algum controle de sessão dos usuários.

² *Alice: Learn Programming*, iniciado em 1999 na Carnegie Mellon University. URL: <http://www.alice.org>.

³ Um *applet* é um mini-aplicativo *Java* (<http://java.sun.com>) que pode ser utilizado dentro de qualquer navegador.

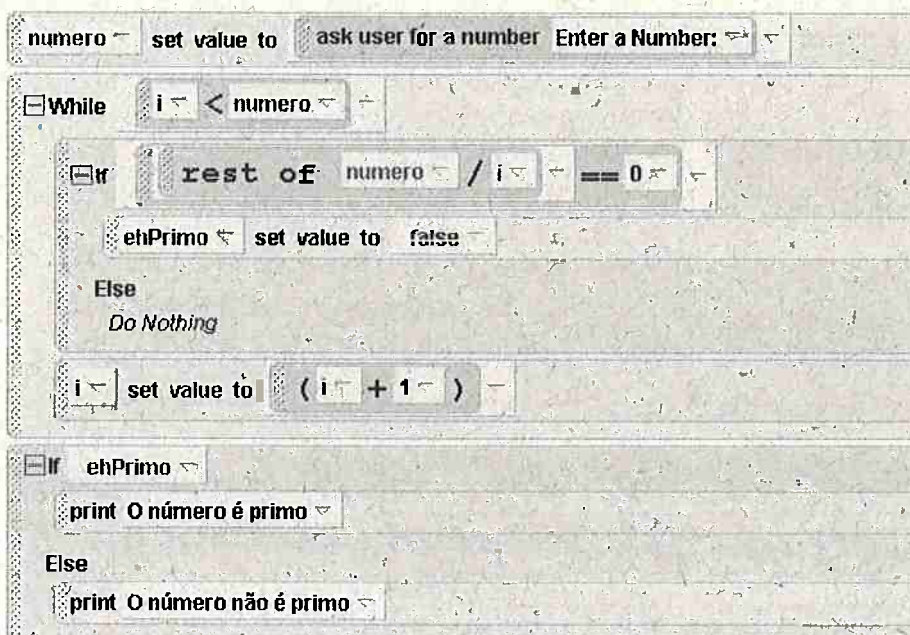


Figura 1.1: Exemplo de algoritmo utilizando a *programação visual*

que por vezes impedem o aluno de conseguir implementar um algoritmo.

As dificuldades relatadas na literatura para se introduzir o conceito de programação deu origem a vários sistemas de apoio, como:

- Portugol* (<http://orion.ipt.pt/manso/Portugol>),
- BlueJ* (<http://www.bluej.org>),
- Jeliot* (<http://cs.joensuu.fi/jeliot>),
- VisuAlg* (<http://www.apoioinformatica.inf.br>),
- Ambap* (<http://www.ufal.br/tci/ambap>),
- ToonTalk* (<http://www.toontalk.com>),
- GameMaker* (<http://www.gamemaker.nl>),
- WebPortugol* (<http://www.univali.br/webportugol>),
- Scratch* (<http://scratch.mit.edu>) e
- Alice* (<http://www.alice.org>).

Destes sistemas apenas o SCRATCH (2010) e o ALICE (2010) seguem o modelo visual, entretanto nenhum dos dois funciona via Web. Mas ambos os sistemas disponibilizam uma versão *Java* que permite executar algoritmos elaborados nos sistemas, reconhecendo a importância atual da utilização destes em navegadores.

1.2 Aprendizagem via Web

Hoje nota-se um crescente uso da Web para apoiar o ensino-aprendizagem no Brasil e no mundo. Consequentemente o aumento na demanda por pesquisas nesta área tem evidenciado algumas características desta abordagem em relação ao método tradicional de ensino (LITTO & FILATRO, 2004).

Por exemplo, a Secretaria Estadual de Educação de São Paulo inicia neste ano dois importantes projetos de uso da Web para formação de professores. O projeto *Escola de Formação de Professores*⁴ e o *Redefor* (Rede São Paulo de Formação Docente)⁵. O primeiro já foi iniciado neste ano, também para capacitação continuada, e está sendo usado como etapa inicial para ingresso na carreira.

Os sistemas computacionais de ensino a distância permitem a interação entre professores e alunos, não só distantes fisicamente, mas também distantes temporalmente, não havendo a necessidade de que esta interação ocorra de forma sincronizada. Esta mudança de contexto desloca a atividade de ensino que conhecemos para um plano mais pessoal, possibilitando que o aprendiz siga seu próprio ritmo de aprendizado no local e horário que lhe for mais conveniente.

Quando a Web passa a ser parte importante também da avaliação da aprendizagem, faz-se necessário o uso de mecanismos de controle de atividades. Para isso existem os *Sistemas Gerenciadores de Cursos (SGC)*. Esta é uma das razões para o grande crescimento do uso de SGC.

Neste sentido destaca-se o sistema *Moodle* (MOODLE, 2010), distribuído na forma de *software livre*, que além de ser o SGC com maior quantidade de usuários, apresenta várias características interessantes, como por exemplo, uma arquitetura que facilita a incorporação de novos recursos, denominados internamente por *módulos*, mas que na área de *engenharia de software* poderiam ser denominados *componentes de software*. Este conceito de componente é um dos grandes propulsores do desenvolvimento contínuo do *Moodle*, existindo atualmente várias centenas deles⁶. Isso significa que, respeitando-se alguns padrões, é possível para qualquer programador incluir um novo módulo (ou adaptar um existente), que atenda às suas necessidades.

⁴ <http://www.escoladeformacao.sp.gov.br>.

⁵ Cujá inscrição esteve aberta para professores da rede estadual até 10/08/2010, mas o sítio oficial até o momento desta redação não está disponível no endereço <http://www.escoladeformacao.sp.gov.br/redefor>.

⁶ Em consulta à página <http://download.moodle.org/stats.php>, em primeiro de setembro de 2010, encontramos 231 módulos.

Uma ideia interessante para possibilitar a incorporação de ferramentas interativas a SGC, de modo flexível, são os *Módulos de Aprendizagem Interativa (iMA)* (BRANDÃO et al., 2006). Os iMA são sistemas *Java* na forma de *applet*, que podem ser integrados a SGC, recebendo e enviando dados para o servidor. Para que um *applet* possa tornar-se um iMA bastaria ao seu desenvolvedor incorporar dois (ou três) novos métodos, um para receber o conteúdo de um servidor, no qual o SGC se encontra, e outro para enviar a resposta do aluno para o mesmo servidor. Se o *applet* dispuser de um avaliador automático, será necessário um terceiro método público, que será consultado pelo servidor via *JavaScript* para definir o resultado da resposta do aluno.

As principais características desses módulos são: ampliação da interatividade ao incorporar recursos para conteúdos específicos, como a programação visual, e disponibilizar recursos de comunicação com um servidor Web através do protocolo *HTTP*. Desta maneira, o servidor pode anotar os dados enviados pelo *iMA* em um banco de dados para que o professor possa examinar as soluções enviadas pelos alunos. Entretanto, notamos uma deficiência no uso de recursos interativos integrados a ambiente Web como aponta RODRIGUES & BRANDÃO (2009). Por exemplo, na versão atual do projeto *Escola de Formação de Professores*, a interatividade do usuário com o conteúdo resume-se a descarregar textos, preencher questionários tipo múltipla escolha e participar de fóruns e *chats*.

1.3 Delimitação do problema

Durante o desenvolvimento deste trabalho buscou-se solucionar os problemas *d1* (dificuldade com o ambiente para programação) e *d2* (dificuldade com a sintaxe) mencionados anteriormente e, como resultado desta busca, é apresentado um novo sistema, o *iVProg*, para *programação visual*, integrável a ambientes Web. Também foram realizados alguns experimentos em disciplina de introdução à computação com esse novo sistema, que comprovaram sua eficácia.

Outro requisito importante deste projeto é possibilitar sua fácil integração a SGC. Como estudo de caso, o *iVProg* foi integrado ao sistema *Moodle* (MOODLE, 2010), a partir de um módulo implementado em nosso grupo de pesquisa (RODRIGUES & BRANDÃO, 2009).

Para permitir uma evolução mais rápida do *iVProg*, seu código será distribuída na forma de *software livre* (RAYMOND, 2000).

A meta principal desse sistema é motivar os estudantes a assimilarem melhor o conceito de

algoritmo, reduzindo problemas típicos dos ambientes tradicionais para ensino-aprendizagem de programação. Além disso, para verificar a validade desta proposta, o iVProg foi testado em aulas de introdução à programação.

1.4 Justificativas

Existem duas justificativas básicas para esse projeto: uma é a produção de ferramentas que catalizem o processo de ensino-aprendizagem via Web e a outra é melhorar o aprendizado específico de programação.

1.4.1 Uso via Web

Apesar da existência de diversos trabalhos bem sucedidos no desenvolvimento de sistemas para ensino de programação como, por exemplo, os sistemas BLUEJ (2010) e JELIOT (2010), existe uma grande lacuna de trabalhos que tenham como objetivo viabilizar o ensino deste tópico através da internet, em cursos presenciais ou a distância. Isso porque são poucos os programas que podem ser executados diretamente em uma página de Web.

1.4.2 Uso da programação visual

A literatura relata várias dificuldades para se introduzir o conceito de programação ou de algoritmos, apontando os problemas que levam os alunos à desmotivação durante o processo de aprendizagem. Entre os vários motivos, estão os erros de sintaxe e compilação durante a simulação dos algoritmos criados pelos alunos (BAEZA, 1995).

Dentro deste cenário, a *programação visual* reduz as dificuldades dos alunos nas disciplinas introdutórias de programação. Dos sistemas anteriormente mencionados, apenas o SCRATCH (2010) e o ALICE (2010) seguem esse modelo.

1.5 Contribuições

Esta dissertação propõe como estratégia de ensino de programação apoiar-se num modelo visual, para reduzir dificuldades típicas enfrentadas por um aluno iniciante na *arte de*

programação, principalmente as dificuldades de configuração de ambiente para programação e da necessidade de memorização de comandos e sintaxe. Além disso propõe-se eleger como prioritário o uso da Web como veículo para promover uma maior participação do aluno em seu aprendizado.

Em função da inexistência de um sistema que atendesse a esses requisitos, foi desenvolvido um novo sistema, o iVProg, baseado num sistema livre pré-existente, o *Alice*. Devido ao tamanho do *Alice* ser impraticável hoje para um *applet*, boa parte do código original do *Alice* precisou ser eliminado (toda a parte de animações 3D), foram refatoradas várias outras partes e ainda incorporadas novas funcionalidades.

Essas novas funcionalidades possibilitam o uso do iVProg de modo integrado a gerenciadores de cursos via Web, como o *Sistema de Aprendizagem pela Web (SAW)* (BRANDÃO et al., 2006) e o *Moodle*.

1.6 Objetivos

O objetivo principal deste trabalho foi desenvolver e testar um novo sistema que funcione via Web e que adote um modelo visual de programação, resultando no sistema iVProg.

Busca-se, por um lado, providenciar recursos que auxiliem o professor na produção de atividades relacionadas à programação básica e ao acompanhamento de seus alunos, e por outro lado, trazer facilidades para um aluno adquirir os conhecimentos nesta área.

No iVProg, a meta foi desenvolver um recurso que permitisse integrá-lo a qualquer sistema gerenciador de curso (*SGC*), implementando o conceito de *Módulos de Aprendizagem Interativa (iMA)* (MOURA, 2007). Já foram feitos testes com uma versão do *Moodle* (RODRIGUES & BRANDÃO, 2009), mas a simplicidade deste recurso permite que iVProg possa ser integrado a qualquer outro *SGC* (vide subseção 3.2).

1.7 Organização do Trabalho

Este texto está estruturado em 5 capítulos.

O Capítulo 2 apresenta os trabalhos relacionados ao ensino de programação divididos em duas partes: novos métodos e novos sistemas.

O Capítulo 3 apresenta o sistema iVProg, descrevendo um pouco sobre seu desenvolvimento e alguns de seus principais recursos disponíveis. Apresenta também o Modelo Visual como metodologia de ensino de programação orientado ao uso componentes gráficos.

O Capítulo 4 apresenta os resultados dos experimentos feitos com o iVProg.

Por último, o Capítulo 5 contém algumas conclusões sobre esta dissertação, discute os resultados alcançados e propostas para futuras implementações ou extensões.

Capítulo 2

Trabalhos relacionados

Existem muitas iniciativas visando a melhoria do ensino-aprendizagem de programação, que podem ser separadas em duas linhas: novos métodos e novos sistemas. No primeiro caso, tenta-se propor novas abordagens, usando-se os mesmos sistemas, enquanto na segunda procura-se desenvolver novos ambientes para programação.

Nas próximas seções são abordadas estas duas linhas.

2.1 Novos métodos

Nessa primeira linha o trabalho de RAPKIEWICZ (2006) descreve a falta de motivação dos alunos e a dificuldade destes em desenvolver o raciocínio lógico necessário para a construção dos algoritmos. Dentro deste contexto, Rapkiewicz propõe estratégias pedagógicas utilizando jogos computacionais para amenizar os problemas citados. Já Borges (2000), propõe como metodologia alternativa utilizar orientação a objetos.

Segundo RAABE et al. (2007) e CAMPOS (2009) é possível observar nos trabalhos desenvolvidos pela comunidade acadêmica brasileira que existe uma tendência para o uso de pseudo-linguagens com palavras reservadas em idioma português para a introdução das noções fundamentais de programação. Rabbe acredita que o uso de português estruturado e de representações visuais (fluxogramas) contribuem para redução dos problemas de aprendizagem na disciplina. Segundo ele, métodos de ensino que exploram estas duas abordagens indicam melhoria na qualidade de aprendizagem e puderam ser mensurados por meio de experimentos realizados com as ferramentas e no índice de reprovação que ao longo dos seis

anos passou de 50% para 35% aproximadamente.

Já MARCZAK & GIRAFFA (2003) propõem como metodologia, o uso de novos Ambientes Inteligentes para Suporte ao Ensino de Programação. Segundo Marczak, a tendência observada nestes ambientes é utilizar técnicas de Inteligência Artificial (IA), a fim de prover a esses sistemas a capacidade de adaptação ao contexto e de personalização do ambiente de acordo com as características dos alunos, além de possibilitar um alto grau de interatividade entre o ambiente e os usuários. A introdução de técnicas de IA nestes ambientes tem, também, a finalidade de propiciar mecanismos de modelagem do processo de ensino, assim como ter a possibilidade de inferir o provável estado cognitivo corrente do estudante. Marczak destaca a preocupação em disponibilizar os conteúdos relacionados ao curso/disciplina que atendem a necessidade do aluno.

Por outro lado, segundo SANTOS & COSTA (2006), no estudo de linguagens de programação, deve ser dada ênfase aos seus aspectos funcionais e estruturais, em detrimento aos detalhes de sintaxe. Conceitos como o significado de associação, avaliação, atribuição, chamada de procedimento, envio de mensagens, passagem de parâmetros, herança, polimorfismo e encapsulamento devem ser enfatizados. O estudo de linguagens de programação deve ser precedido do estudo dos principais paradigmas de programação, notadamente a programação imperativa, a funcional, a baseada em lógica e a orientada a objetos. Conforme destaca Santos, deve-se entender que ensinar programação não é simplesmente ensinar uma linguagem de programação. Esse ensino envolve sobretudo entender problemas é descrever formas de resolução, de maneira imparcial, para que então sejam codificadas em uma linguagem de programação.

Como nova metodologia, PEREIRA (2005) propõe o ensino de algoritmos e programação nas escolas do nível médio, antecipando o conhecimento nesta área por parte dos alunos. Segundo o autor, entre as competências mais difíceis de serem desenvolvidas estão as relacionadas com o desenvolvimento de algoritmos e programas. Essa é uma forte razão para incluir tal temática no ensino médio - assim, os futuros alunos dos cursos de computação teriam menos dificuldade nessa área. E aqueles que se dirigissem para outros ramos profissionais teriam desenvolvido competências de resolução de problemas e formalização dos mesmos que seriam úteis em suas respectivas áreas do conhecimento.

Segundo outra metodologia, WESLEY et al. (2008) propõem o uso de fluxogramas no ensino de algoritmos. Segundo Wesley, o uso de fluxogramas se deve a três razões principais. Primeiro, fluxogramas possuem uma sintaxe mínima. Quando se reduz o foco em sintaxe, pode-se aumentar o esforço em análise. Segundo, fluxograma é uma representação universal.

Nenhum outro sistema visual alcançou a aceitação dos fluxogramas. Terceiro, fluxogramas são mais fáceis de entender para estudantes iniciantes em computação do que estrutura de código. Wesley afirma também que apesar do fluxograma ser uma das formas mais simples de se observar, criar e apresentar um algoritmo em forma gráfica, o uso de ferramentas computacionais tradicionais reduz a naturalidade e a liberdade do usuário na expressão de suas ideias se comparado ao uso de lápis e papel.

Todos esses trabalhos evidenciam que é grande a preocupação por melhorias no ensino de programação. Vale destacar que o iVProg pode ser utilizado principalmente no projeto de PEREIRA (2005), aprimorando a metodologia de ensino de programação em escolas do ensino médio.

2.2 Novos sistemas

Diversos sistemas foram desenvolvidos no ensino de programação, dos quais poucos seguem um modelo que reduza os problemas relacionados à compilação e sintaxe, mas serviram como primeiro impulso para o progresso educacional na área. Se são raros os sistemas que atendam o requisito citado, mais raros ainda são os que possuem a funcionalidade de edição disponível na internet.

Na pesquisa dos trabalhos relacionados, foram analisadas essas características além da disponibilização gratuita e livre do código fonte (RAYMOND, 2000), ou seja, possibilitar o uso da totalidade ou parte do código fonte para modificações e a sua distribuição. Essa condição é fundamental para possibilitar a adaptação do sistema. A tabela 2.1 destaca as características dos principais sistemas pesquisados.

Tabela 2.1: Análise dos Sistemas para Ensino de Programação.

<i>Sistema</i>	<i>CW</i>	<i>TW</i>	<i>PV</i>	<i>CL</i>	<i>URL</i>
Portugol					http://orion.ipt.pt/manso/Portugol/
BlueJ					http://www.bluej.org
Jeliot					http://cs.joensuu.fi/jeliot/
VisuAlg					http://www.apoioinformatica.inf.br
Ambap					http://www.ufal.br/tci/ambap
ToonTalk					http://www.toontalk.com/
GameMaker					http://www.gamemaker.nl
WebPortugol	X	X			http://www.univali.br/webportugol

Scratch		X	X	X	http://scratch.mit.edu/
Alice		X	X	X	http://www.alice.org/

Na tabela 2.1, a coluna “CW” aponta os sistemas que possuem recursos para criação ou edição de programas/algoritmos na Web. A coluna “TW” destaca aqueles que possuem um “tocador” na Web, ou seja, permitem ao usuário apenas visualizar a execução na Web mas não necessariamente a criação e edição de programas/algoritmos na internet. A coluna “PV” mostra os sistemas que utilizam a programação visual, é importante destacar que neste requisito estamos considerando os sistemas nos quais o modelo visual é usado na construção do programa/algoritmo e o usuário não precisar digitar os comandos. A coluna “CL” destaca aqueles que possuem um *código livre*, desta maneira permitem o uso da totalidade ou parte do código fonte para modificações e a sua distribuição independente da licença. A coluna “URL” mostra o endereço no qual o sistema pode ser acessado ou descarregado.

2.2.1 Sistemas de ensino *off-line*

Foram classificados como sistemas de ensino *off-line* os que não possuem recursos para criação ou edição de programas/algoritmos na Web nem um “tocador” na Web.

A maioria dos sistemas funcionam basicamente no modo aplicativo e não disponibilizaram funcionalidades na Web, mas tiveram grande contribuição para o processo educacional na área. A seguir, são apresentadas as características dos principais sistemas pesquisados nesta categoria.

Portugol

Portugol é um simulador de linguagem algorítmica desenvolvido para apoio às aulas de Introdução à Programação dos Cursos de Engenharia da Escola Superior de Tecnologia do Instituto Politécnico de Tomar (MANSO, 2010).

A linguagem algorítmica utiliza o português para a definição dos comandos e tem as seguintes características:

- A linguagem não faz distinção entre letras minúsculas e maiúsculas
- Cada linha contém apenas um comando

- O sinal de atribuição é a seta
- A definição de símbolos é feita em qualquer local do algoritmo

Não possui editor de algoritmos na internet, nem utiliza a programação visual.

Maiores informações estão disponíveis no site oficial <http://orion.ipt.pt/manso/Portugol>

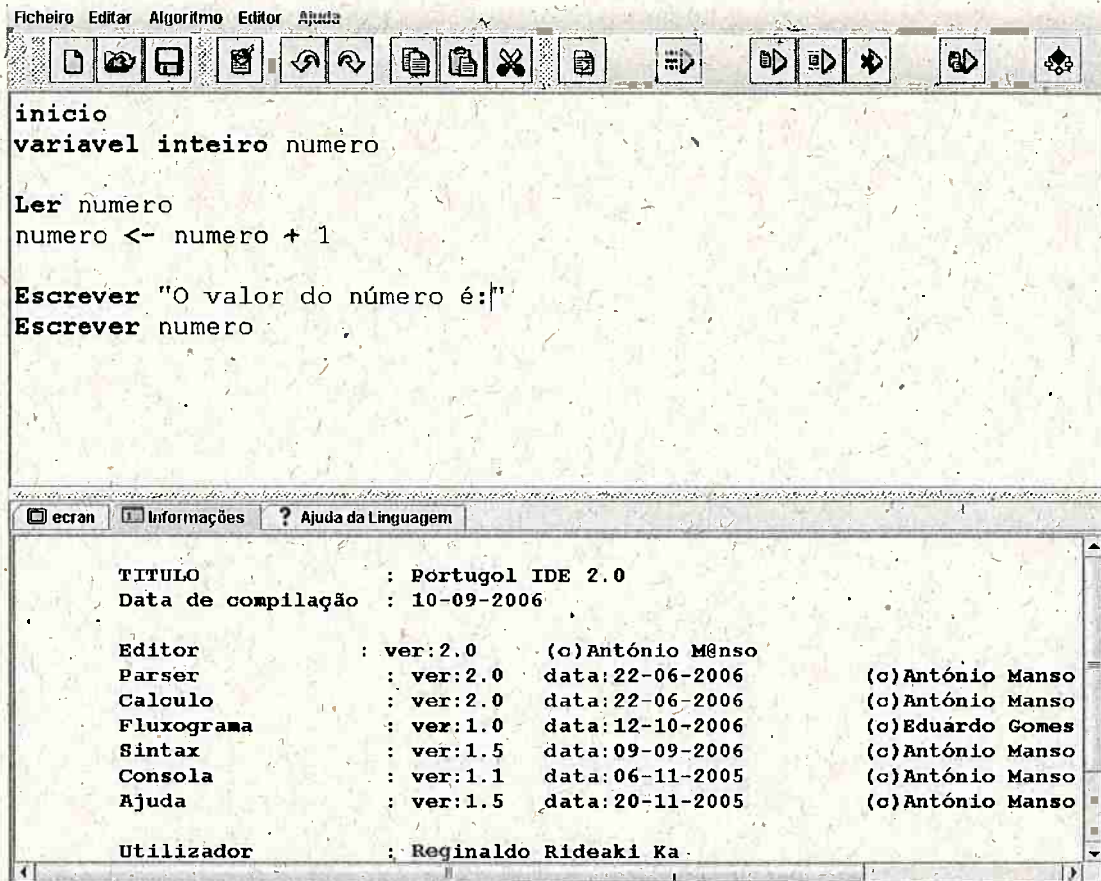


Figura 2.1: Tela principal do Portugol.

BlueJ

BlueJ é um ambiente de desenvolvimento java projetado especificamente para o ensino em um nível introdutório. BlueJ foi projetado e implementado pelas equipes das universidades Monash University, Melbourne, Australia e The University of Southern Denmark, Odense (BLUEJ, 2010).

Ambiente integrado desenvolvido para trabalhar com elaboração de projetos em Java, o BlueJ foi desenvolvido com base no Blue System, que é um software elaborado pela Universidade de Sidney com o intuito de auxiliar os alunos no processo de aprendizagem em programação. É adequado para quem está iniciando no trabalho com a linguagem de programação Java. Possui um ambiente interativo, no qual é possível visualizar as classes através de um diagrama de classes. A figura 2.2 mostra um exemplo com o diagrama de classes.

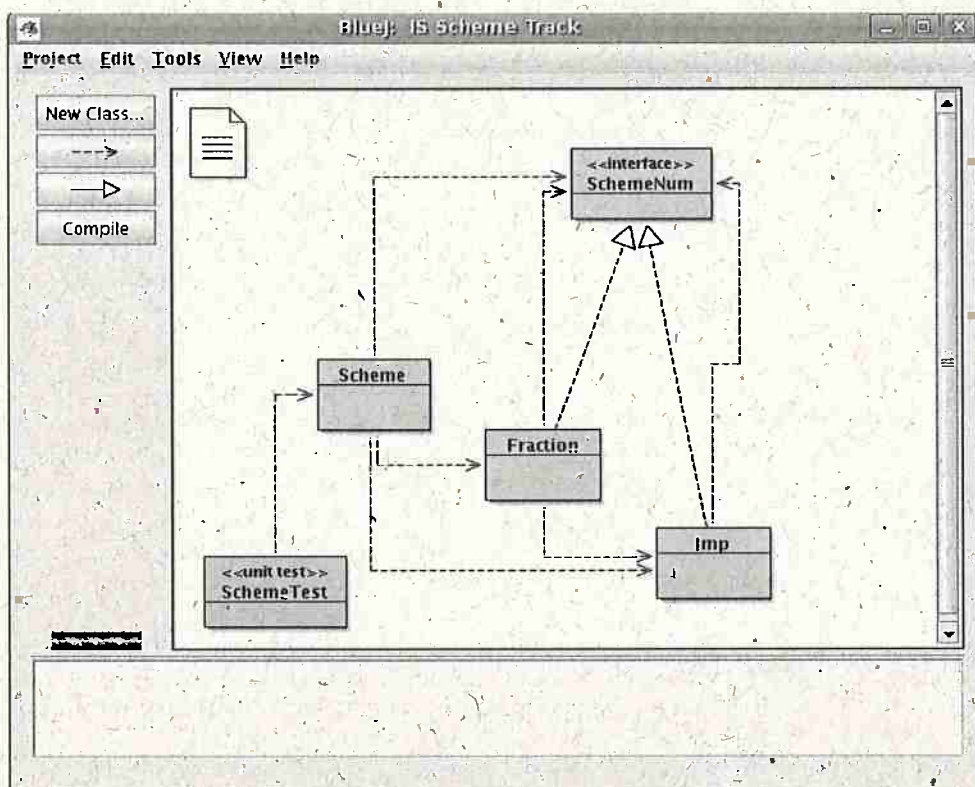
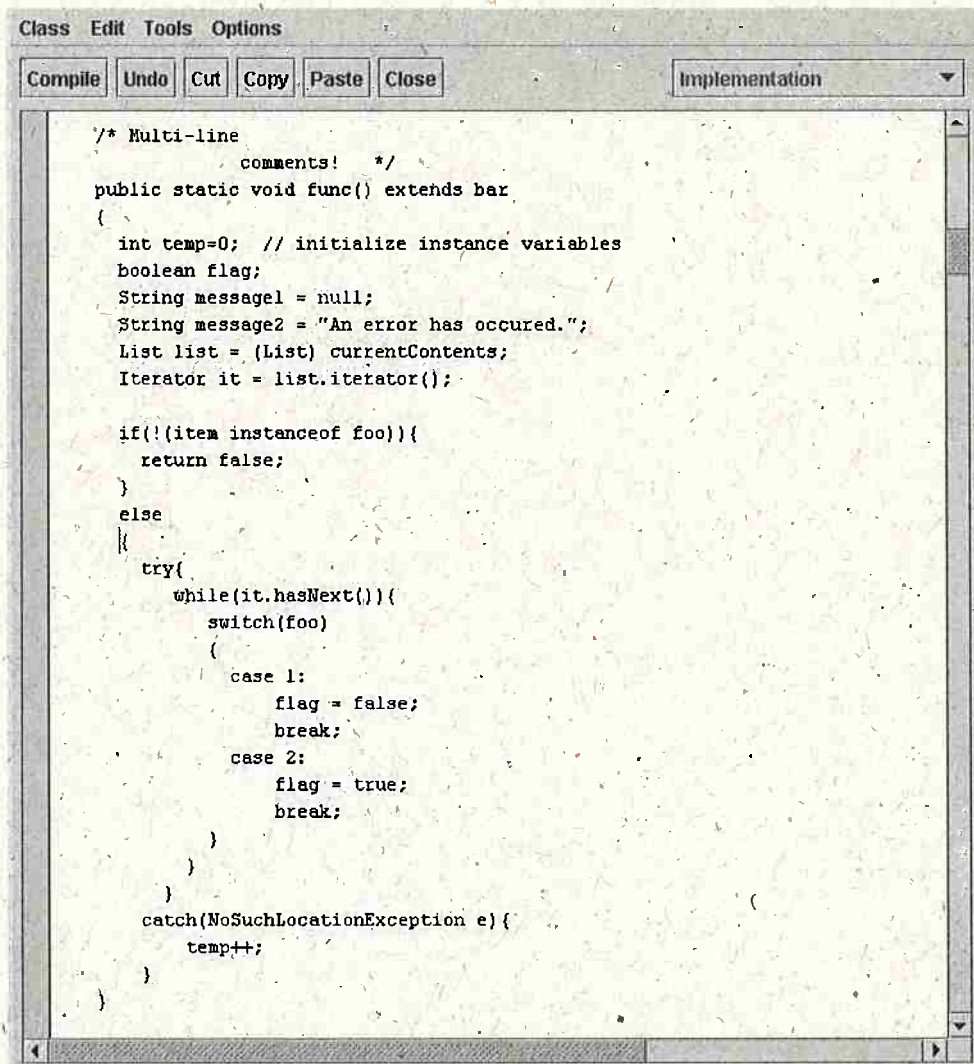


Figura 2.2: Exemplo de algoritmo no BlueJ.

Já a figura 2.3 mostra um exemplo de algoritmo digitado no Bluej.



The image shows a screenshot of a BlueJ IDE window. The window title is "Class Edit Tools Options". Below the title bar, there are buttons for "Compile", "Undo", "Cut", "Copy", "Paste", and "Close", along with a dropdown menu set to "Implementation". The main text area contains the following Java code:

```
/* Multi-line
   comments! */
public static void func() extends bar
{
    int temp=0; // initialize instance variables
    boolean flag;
    String message1 = null;
    String message2 = "An error has occurred.";
    List list = (List) currentContents;
    Iterator it = list.iterator();

    if(!(item instanceof foo)){
        return false;
    }
    else
    {
        try{
            while(it.hasNext()){
                switch(foo)
                {
                    case 1:
                        flag = false;
                        break;
                    case 2:
                        flag = true;
                        break;
                }
            }
        }
        catch(NoSuchLocationException e){
            temp++;
        }
    }
}
```

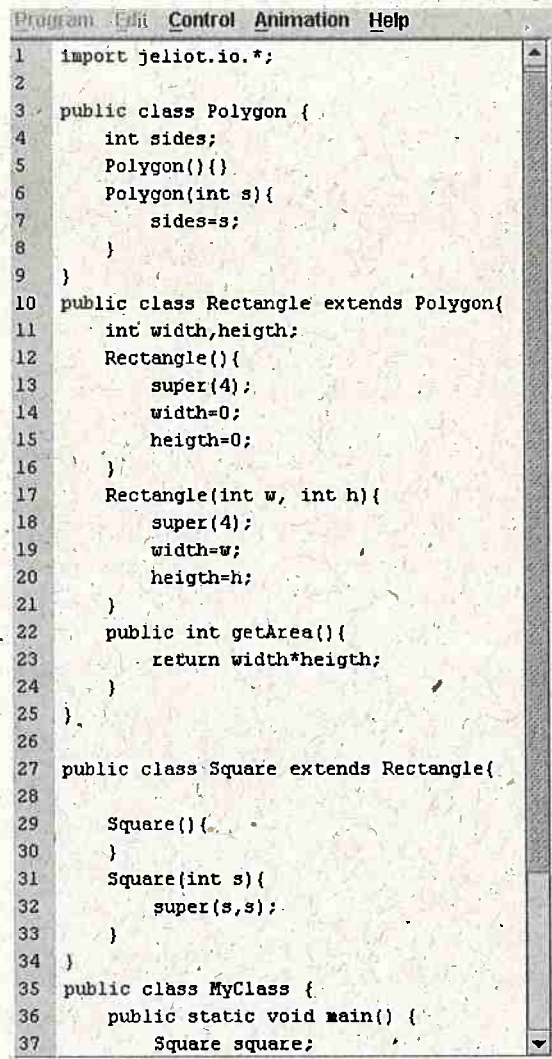
Figura 2.3: Exemplo de algoritmo no BlueJ.

Jeliot

Jeliot é um sistema voltado para o ensino de programação orientada a objetos com Java. Neste ambiente, é possível fazer simulações gráficas da execução dos algoritmos, o que permite que se veja como os programas são interpretados (JELIOT, 2010).

Durante a execução dos programas no Jeliot, é possível ver: as variáveis sendo declaradas e inicializadas; as expressões sendo avaliadas; os objetos sendo construídos; os métodos sendo chamados;

Maiores informações estão disponíveis no site oficial <http://cs.joensuu.fi/jeliot/>.



```
Program Edit Control Animation Help
1  import jeliot.io.*;
2
3  public class Polygon {
4      int sides;
5      Polygon(){}
6      Polygon(int s){
7          sides=s;
8      }
9  }
10 public class Rectangle extends Polygon{
11     int width,height;
12     Rectangle(){
13         super(4);
14         width=0;
15         height=0;
16     }
17     Rectangle(int w, int h){
18         super(4);
19         width=w;
20         height=h;
21     }
22     public int getArea(){
23         return width*height;
24     }
25 }
26
27 public class Square extends Rectangle{
28     Square(){
29     }
30     Square(int s){
31         super(s,s);
32     }
33 }
34 }
35 public class MyClass {
36     public static void main() {
37         Square square;
```

Figura 2.4: Tela do Jeliot.

VisuAlg

O VisuAlg é um editor e interpretador de pseudocódigos que possibilita aos alunos iniciantes em programação o exercício dos seus conhecimentos (VISUALG, 2010).

A ferramenta é capaz de depurar a execução dos códigos, ou seja, possibilita a verificação dos valores das variáveis e o acompanhamento da execução de um algoritmo (VISUALG, 2010).

Maiores informações estão disponíveis no site oficial

<http://www.apoioinformatica.inf.br/visualg/telaprin.htm>.

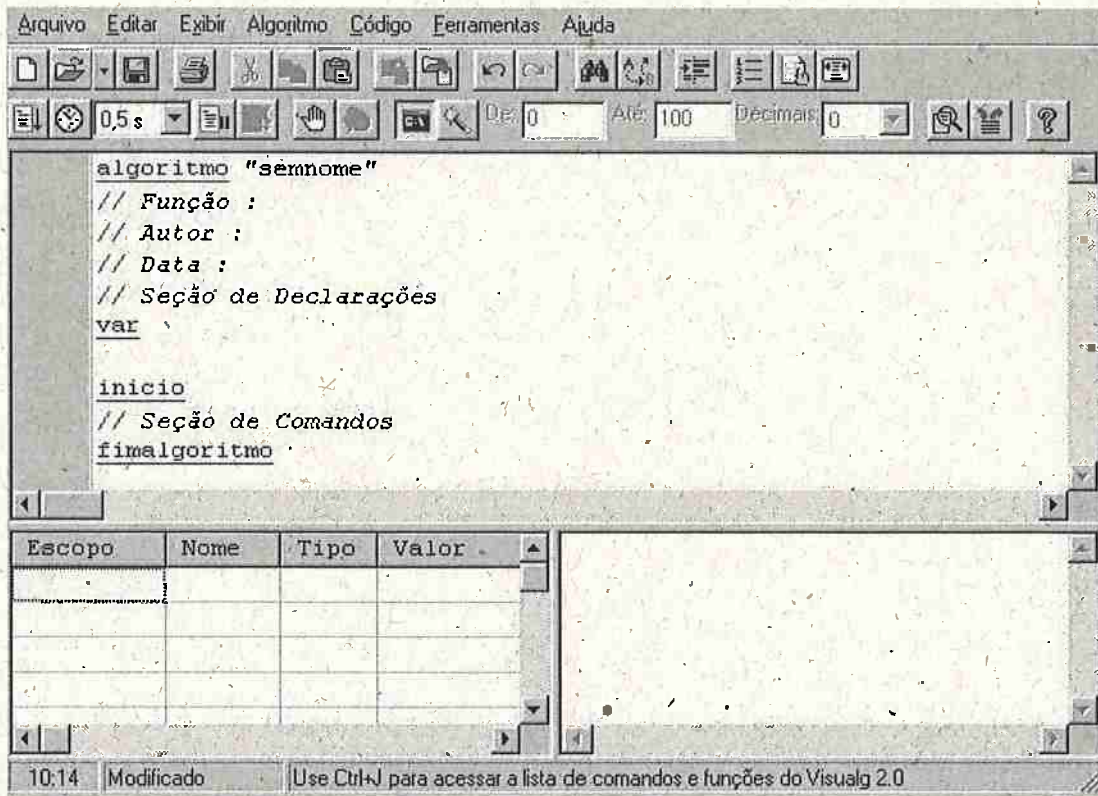


Figura 2.5: Tela principal do VisuAlg.

AMBAP

AMBAP (Ambiente de Aprendizado de Programação) é um ambiente desenvolvido pelos alunos de ciência da computação da Universidade Federal de Alagoas.

O ambiente facilita o aprendizado de lógica de programação, uma vez que utiliza que utiliza linguagem de programação, com comando em Português, denominado português estruturado (AMBAP, 2010).

Maiores informações estão disponíveis no site oficial <http://www.ufal.br/tci/ambap>.

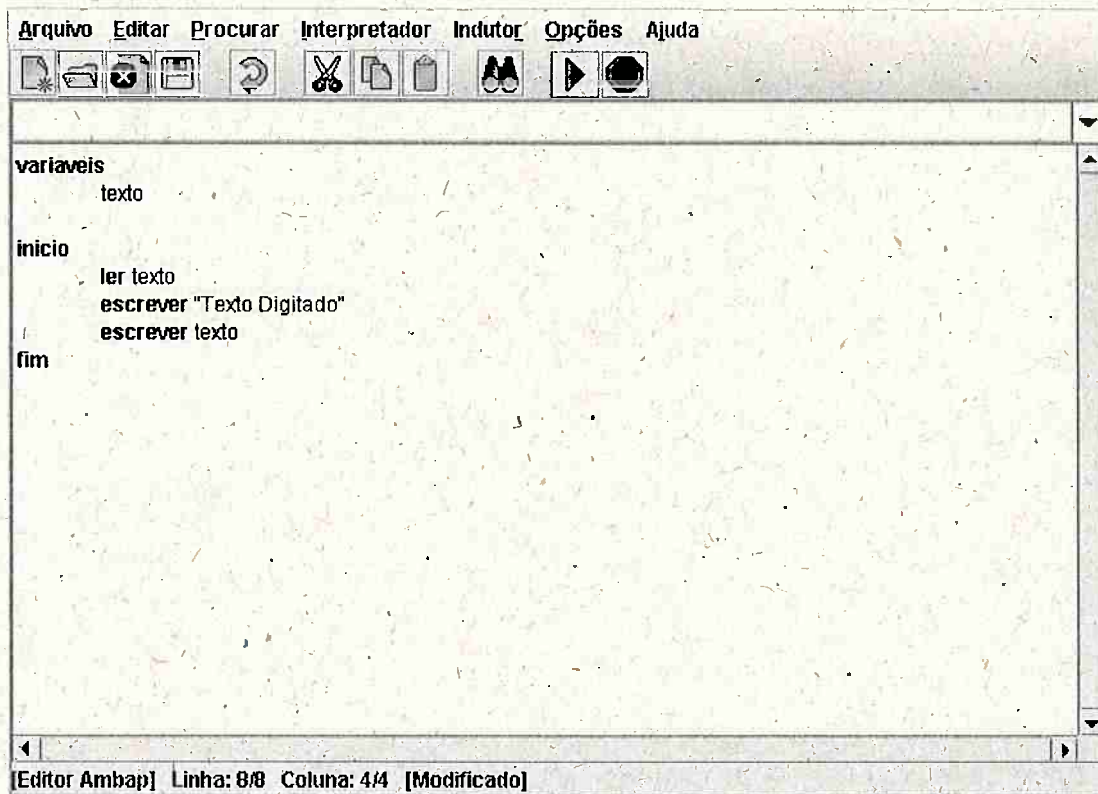


Figura 2.6: Tela do AMBAP.

ToonTalk

ToonTalk é um ambiente de programação voltado exclusivamente para crianças e adolescentes. Trata-se de uma ferramenta orientada a objetos, em que a criança pode projetar, estruturar e rodar seus programas (Kahn, 2010).

Todo o processo de aprendizagem ocorre em um mundo virtual tridimensional, no qual personagens animados interagem com a criança. Assim, ela pode desenvolver sua capacidade de solucionar problemas, a criatividade e o raciocínio abstrato.

O software permite também que a criança crie seus trabalhos em Java e coloque-os na própria página Web pessoal. Entretanto, não possui editor de algoritmos na internet, apenas versão aplicativo executável no Sistema Operacional Windows. Além disso, o código fonte não é livre e o sistema é pago.

Maiores informações estão disponíveis no site oficial <http://www.toontalk.com/>.

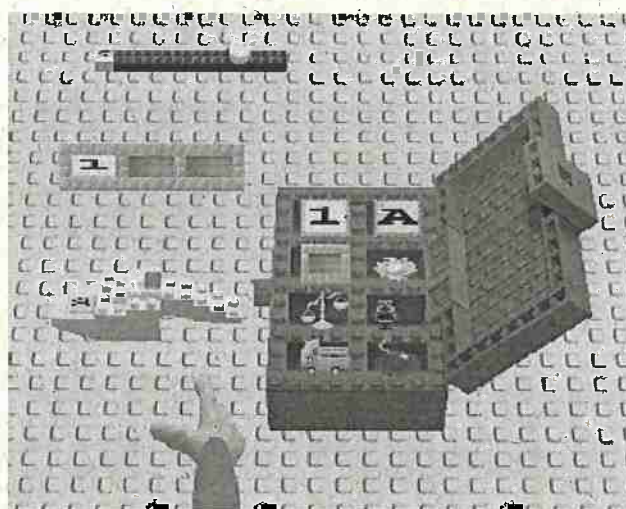


Figura 2.7: Exemplo de criação no Toontalk.

Greenfoot

O Greenfoot, desenvolvido pela Universidade de Kent, é uma ferramenta de software concebida para possibilitar que o iniciante adquira experiência com programação orientada a objeto. Ele suporta o desenvolvimento de aplicações gráficas na Linguagem de Programação Java.

O Greenfoot foi concebido e implementado na Universidade de Kent, Inglaterra e na Universidade Deakin, Melbourne, Austrália.

Maiores informações estão disponíveis no site oficial <http://www.greenfoot.org>.

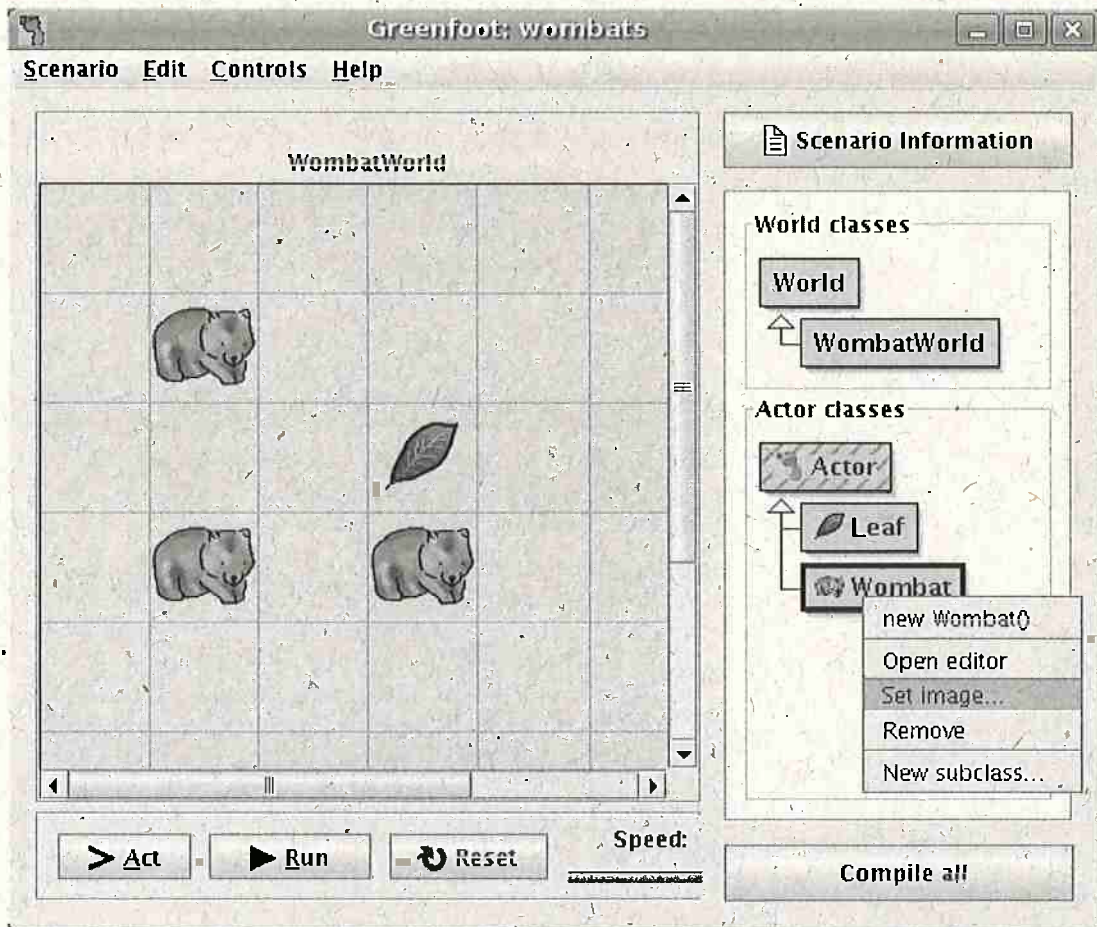


Figura 2.8: Tela principal do Greenfoot.

O Greenfoot utiliza um modelo visual para representar as heranças e as relações entre os objetos, como ilustrado na figura 2.8. Porém, não é utilizada a programação visual, pois há a necessidade de digitar os comandos como mostra o exemplo da figura 2.9.

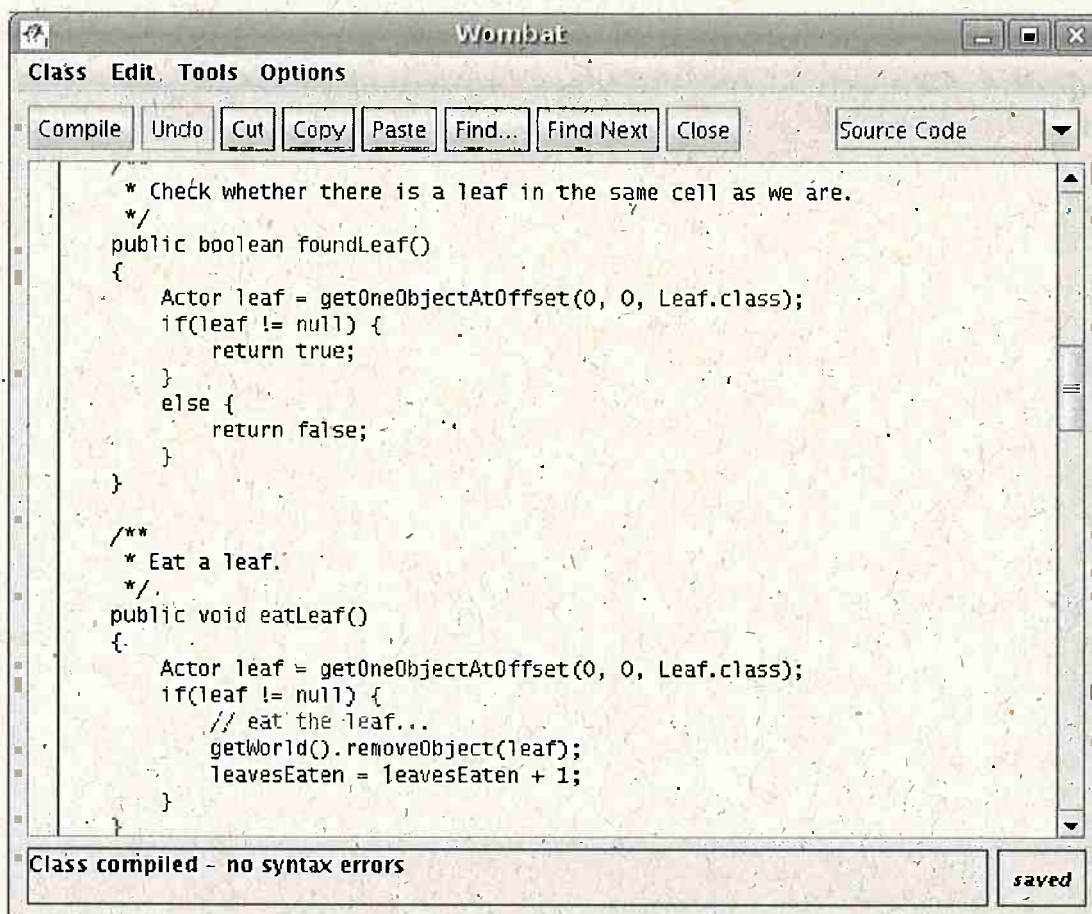


Figura 2.9: Exemplo de edição de algoritmo no Greenfoot.

GameMaker

Game Maker (também conhecido como GM) é um sistema proprietário para programar jogos, desenvolvido pela YoYo Games. O sistema tem suporte a uma linguagem chamada GML.

Todos os recursos dos jogos são organizados em pastas dentro do programa, que inclui pequenos programas para criar seus recursos, como editores de imagens e sons. O Game Maker permite ainda salvar os recursos criados para que possam ser usados em outros jogos ou fora do programa e importar ações adicionais para estender as funções do programa (GAMEMAKER, 2010).

Maiores informações estão disponíveis no site oficial <http://www.gamemaker.nl>.

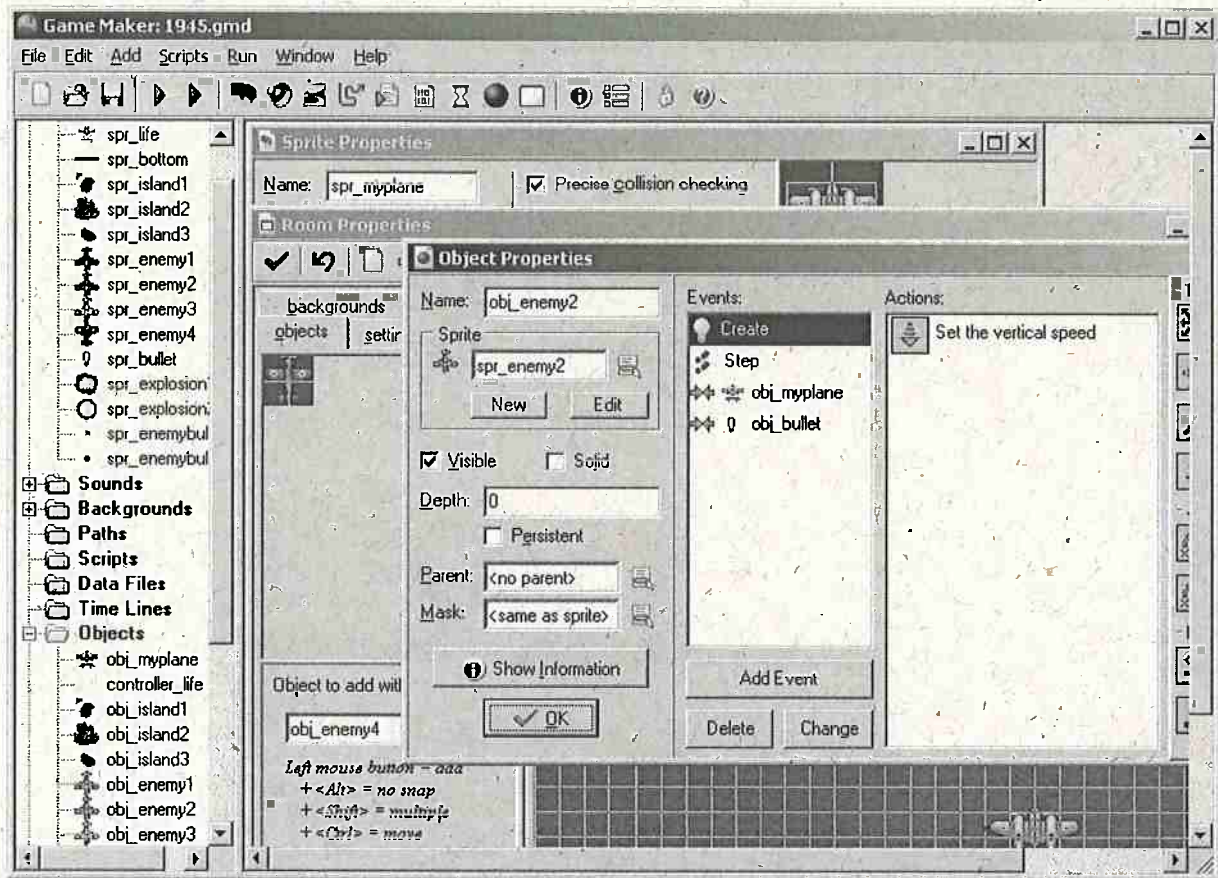


Figura 2.10: Exemplo de criação de um jogo no GameMaker

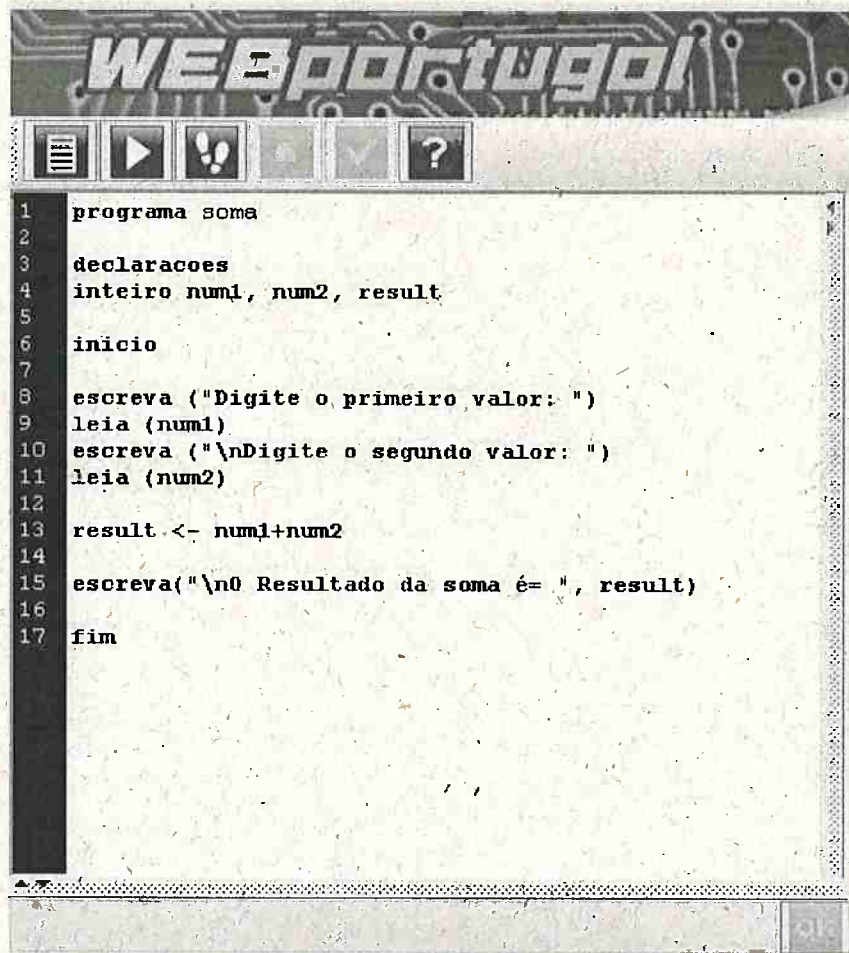
2.2.2 Sistemas de ensino *on-line*

A internet oferece vários recursos que possibilitam o estudo de diversas áreas do saber. Dentre esses recursos, podem ser encontrados alguns sistemas para ensino de programação que possuem a funcionalidade criar programas/algoritmos *on-line* ou um “tocador” na Web. Foram selecionados alguns que atendem ao menos um destes requisitos.

WebPortugol

O Webportugol é uma ferramenta para ajudar os alunos a fazerem seus primeiros algoritmos e com isso aprenderem a lógica de programação. A ferramenta usa a linguagem português que permite criar programas em português (WEBPORTUGOL, 2010).

Maiores informações estão disponíveis no site oficial <http://www.univali.br/webportugol>



```
1 programa soma
2
3 declaracoes
4 inteiro num1, num2, result
5
6 inicio
7
8 escreva ("Digite o primeiro valor: ")
9 leia (num1)
10 escreva ("\nDigite o segundo valor: ")
11 leia (num2)
12
13 result <- num1+num2
14
15 escreva("\n0 Resultado da soma é= ", result)
16
17 fim
```

Figura 2.11: Tela do WebPortugol.

Scratch

Scratch é um sistema desenvolvido pelo Lifelong Kindergarten Group do Massachusetts Institute of Technology (MIT) Media Lab, destinado à criação de jogos simples, animações, histórias interativas e músicas (SCRATCH, 2010).

A sua principal característica é possuir um ambiente de programação que não requer conhecimentos prévios de algoritmos ou de programação para o seu manuseio, pois utiliza programação visual.

Os projetos desenvolvidos no Scratch podem ser compartilhados num site do próprio MIT de forma muito simples, através de um pequeno cadastro. É possível visualizar os jogos e animações na internet, porém a edição de algoritmos é disponibilizada apenas na versão aplicativo.

Por ser destinado a crianças, não possui componentes básicos para construção de programas, como por exemplo, para entrada do usuário, manipulação de string, expressões de lógica, matemática, etc.

Maiores informações estão disponíveis no site oficial <http://scratch.mit.edu/about>



Figura 2.12: Tela principal do Scratch.

Alice

Alice (ALICE, 2010) é um ambiente de programação 3D que facilita a criação de animações e jogos interativos. Permite que os estudantes aprendam conceitos de programação fundamentais de forma lúdica.

Maiores informações estão disponíveis no site oficial <http://www.alice.org/>

Este foi o sistema escolhido para ser adaptado. Todos os fundamentos para essa escolha e a descrição das características do *Alice* estão detalhadas no próximo capítulo.

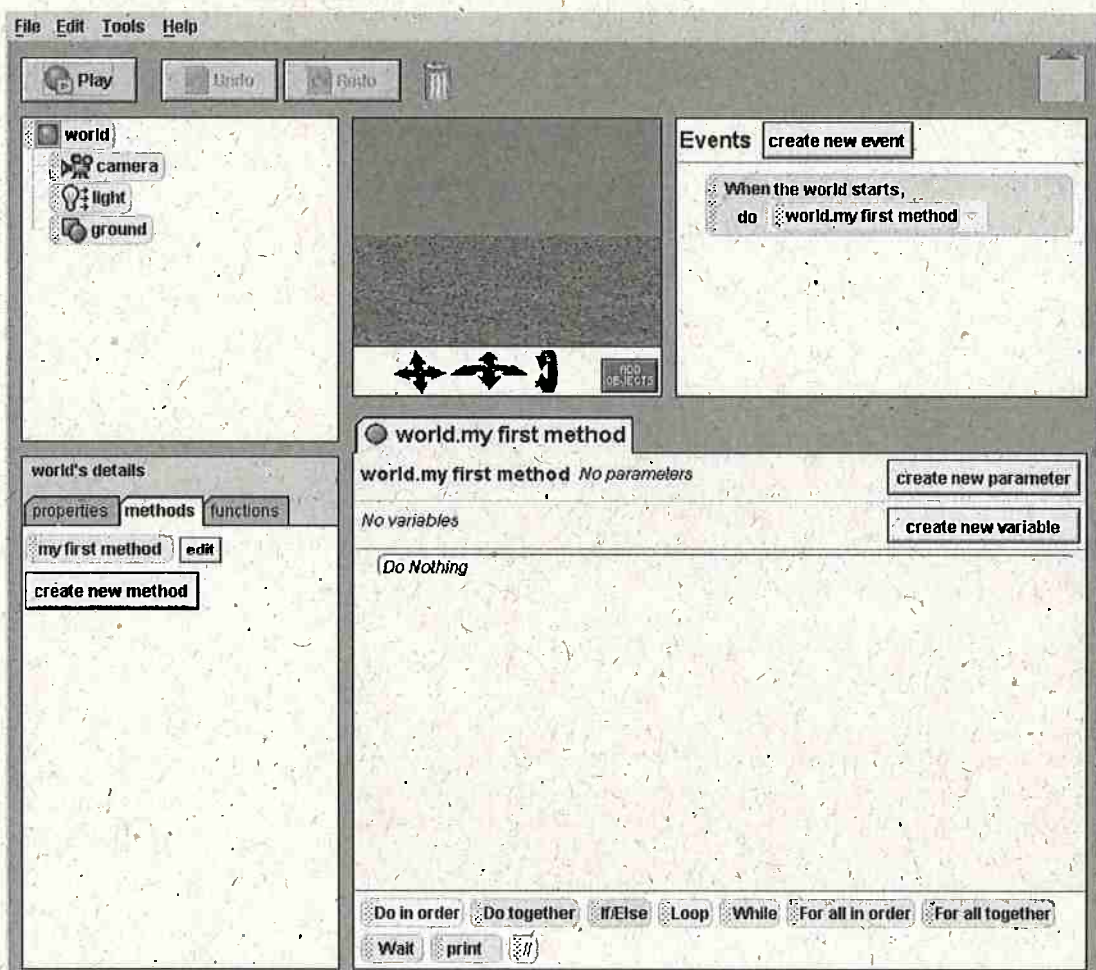


Figura 2.13: Tela principal do Alice (original).

Capítulo 3

O sistema iVProg

O iVProg é destinado aos alunos do ensino técnico ou superior em disciplinas introdutórias de algoritmos/programação e aos alunos do ensino médio em disciplinas cujo objetivo seja a melhoria na compreensão de algoritmos básicos utilizados na matemática.

Este projeto utilizou como base o sistema *Alice*, por este atender a 3 importantes requisitos: ter seu *código livre* (RAYMOND, 2000), ser implementado em *Java* (o que possibilitou portá-lo para a *Web*) e dispor do mecanismo visual para programação.

Entretanto foi necessário um grande esforço para viabilizar uma versão do *Alice* que pudesse ser utilizada na *Web* (na forma de *applet*), pois o sistema estava na forma de aplicativo e com um tamanho (em *bytes*) que inviabilizaria sua carga como *applet*: atualmente o *Alice* tem 140 Mb. Para isso foi necessário eliminar todo o código relacionado com animação 3D. Mas, esse corte não provocou prejuízo para os objetivos e público-alvo do iVProg, pois interessava-nos seus recursos gráficos para comandos de programação.

Podemos destacar que o objetivo maior deste trabalho foi alcançado, que era produzir um sistema de apoio ao ensino-aprendizagem de programação. Para isso foram feitas várias adaptações no sistema *Alice*, além de novas implementações, como a internacionalização e o sistema de autoria para página interativas com o iVProg. Estas contribuições serão detalhadas nas próximas seções.

Outra observação que merece destaque é que o iVProg está disponível gratuitamente a partir do sítio do *Projeto iMática*¹, no endereço <http://www.matematica.br/ivprog/>. Brevemente será disponibilizado também o código fonte.

¹ Um projeto para disponibilização de recursos didáticos digitais para apoiar o ensino-aprendizagem de Matemática, disponível na URL: <http://www.matematica.br>.

3.1 Descrição geral do iVProg

O iVProg é um sistema para ensino-aprendizagem de programação, que funciona integralmente no modo *applet* (via Web). Para isso, ele implementa um modo de programação visual, baseado em componentes gráficos para representar comandos. Além disso, incorporamos ao iVProg o conceito de internacionalização, disponibilizando-o inicialmente em Português e Inglês.

Nas próximas subseções, detalharemos estas três características.

3.1.1 Aplicativo/applet

O iVProg funciona tanto como aplicativo quanto como *applet*, dispondo das mesmas funcionalidades, com a restrição usual de segurança para um *applet* acessar ao disco do usuário (SUN, 2010).

No caso aplicativo, implementamos a possibilidade de comandos especiais para abertura do iVProg. Assim, é possível, com linha de comando, iniciá-lo com um arquivo ou com língua específicos, por exemplo, `java -jar iVprog.jar lang=en exemplo1.ivp` abriria o iVProg, com o arquivo `exemplo1.ivp` e usando a língua inglesa.

Para possibilitar o carregamento de arquivos iVProg no modo *applet* foi necessário definirmos uma estrutura interna de representação de conteúdo, usando o formato XML², uma vez que a estrutura de representação de arquivos, original do *Alice*, é um arquivo compactado no formato ZIP³.

Uma vez definida esta estrutura XML, que possibilita o carregamento de arquivos iVProg em página Web, implementamos alguns métodos para melhor integrar o iVProg em formato *applet* a servidores, em particular ao Moodle MOODLE (2010). Estas funcionalidades estão melhor descritas na seção 3.2.

² Abreviação do inglês *eXtensible Markup Language*, é uma linguagem de marcação recomendada pela W3C (<http://www.w3.org/>) para a criação de documentos com dados.

³ Extensão que indica um arquivo compactado no formato ZIP. Um arquivo ZIP contém um ou mais arquivos que foram compactados para reduzir o tamanho do arquivo.

3.1.2 Modelo de programação (visual) do iVProg

Como já discutido na seção 1.1, o modelo de programação visual tem como meta geral facilitar o contato inicial de um aprendiz de programação, permitindo que num curto período de tempo ele seja capaz de fazer e testar alguns programas.

As duas contribuições mais importantes da programação visual estão relacionadas à redução de dificuldades encontradas por alunos em cursos tradicionais de introdução à programação. A primeira é que esse modelo reduz significativamente as dificuldades de alunos com a sintaxe rígida (e não natural) de linguagens típicas que são utilizadas em cursos introdutórios de programação (como Java ou C). A segunda contribuição, é reduzir substancialmente a curva de aprendizado sobre o ambiente de programação, pois como será apresentado a seguir, sua interface permite um rápido aprendizado da sintaxe de programação.

Um programa/algoritmo é construído no iVProg agrupando-se componentes para contextos destacados. No momento em que o aluno clica no componente que deseja utilizar, uma borda verde é mostrada nos locais em que o aluno pode utilizar o comando, indicando ser este um *bloco habilitado* a receber o bloco inicial. Deste modo, o usuário sabe que não tem efeito tentar colocar um bloco qualquer sobre outro que não esteja habilitado.

Um bom exemplo da eficácia deste modelo é evitar um erro comum que alunos iniciantes cometem com linguagens com C ou Java: a diferença proporcionada por um sinal adicional de igualdade, ou seja, $a==b$ (expressão lógica) é muito diferente de $a=b$ (atribuição).

Neste caso, como representado na figura 3.1, pode-se colocar um bloco do tipo $a==b$ (à esquerda na figura) para a linha do bloco Se (segunda linha de comandos, à direita na figura, com a borda verde), que é o contexto correto de uma expressão lógica. Entretanto o iVProg não permite copiar o bloco $a==b$ sobre o bloco de atribuição de valor à variável (em amarelo, na primeira linha à direita na figura).

Na figura 3.1 também é possível observar que cada comando é representado por um bloco com cores distintas, além de automaticamente “identá-los”. Deste modo fica mais fácil o aluno identificar o escopo do comando.

Outra facilidade do iVProg é reduzir o problema de falta de inicialização de variável, uma vez que ao criar uma variável é sempre necessário atribuir um valor inicial. Também ao arrastá-la para a área de trabalho (área maior em azul na figura 3.1, na qual constroem-se os programas), faz-se necessário atribuir algum valor inicial. Neste exemplo, o sistema atribuiu automaticamente o valor 1 para a variável numero.

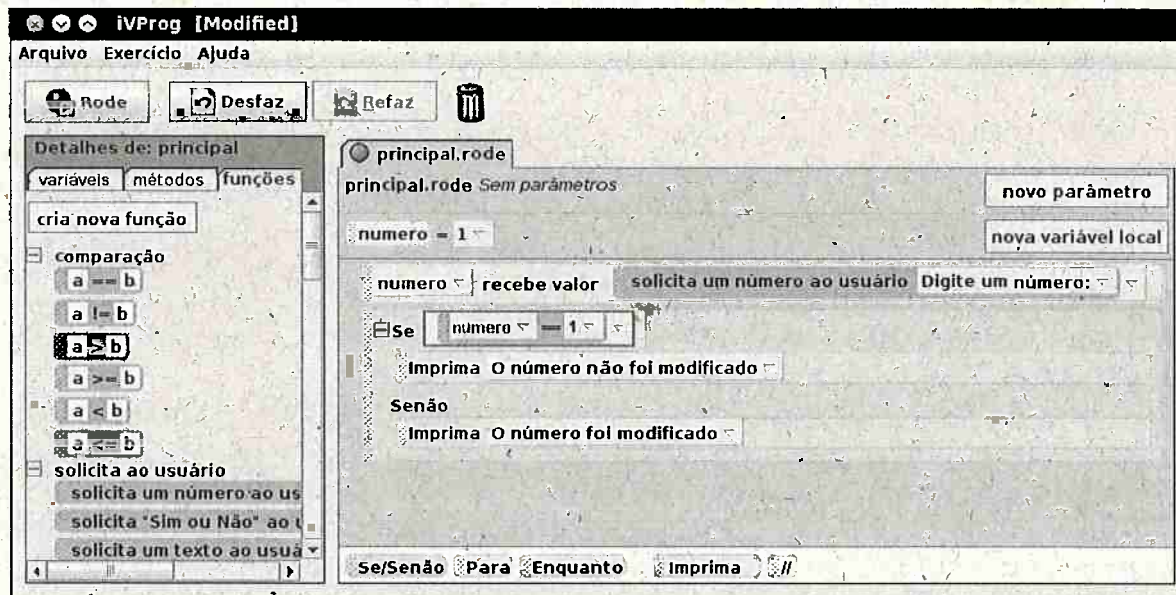


Figura 3.1: Contexto destacado na programação visual do iVProg.

A execução de um programa no iVProg é simples, bastando clicar no botão Rode (canto superior da figura 3.1) e o resultado da execução é apresentado em uma janela específica. A funcionalidade de cada módulo e os detalhes dos componentes gráficos estão descritos no Apêndice A.

É importante destacar que esse modo de operação (“clique e arrastar” blocos), herdado pelo modelo chamado *Drag and Drop* do *Alice*, apresenta uma falha que resulta em funcionamento não uniforme e, portanto, não portátil na versão *applet*. Após diversas análises, descobrimos que existe um erro de permissão oculto pelo código herdado do *Alice* referente ao evento de *Drag and Drop*⁴. Esta falha ocorre apenas na versão *applet*, pois na versão aplicativo o evento de *Drag and Drop* independe da configuração do *Java* ou do sistema operacional.

A solução proposta seria o modelo “clique-pega” e “clique-solta” que resolveria a falha citada. Entretanto ela ainda está em desenvolvimento devido a complexidade do código herdado do *Alice* que está extremamente acoplado ao modelo *Drag and Drop*, exigindo uma grande refatoração.

⁴O evento *Drag and Drop* lança uma exceção de acesso quando executado no modo *applet*, conforme descrito, por exemplo, na URL <http://www.rockhoppertech.com/java-drag-and-drop-faq.html>

3.1.3 Internacionalização

Internacionalização de um sistema é um processo ou princípio de desenvolvimento que procura adaptar um sistema à língua e cultura de um país. Dependendo dos requisitos do sistema em questão é desejado prover uma estrutura de internacionalização capaz de tornar o sistema adaptado em relação a vários aspectos, entre eles: idiomas, formato de data e formatação de unidades monetárias. Enquanto formatação de datas e unidades monetárias pode ser confiada a bibliotecas previamente escritas, a internacionalização referente a mudança de idiomas exige a criação de um arquivo específico para cada idioma desejado, uma vez que esse conhecimento é dependente do domínio, em outras palavras, é dependente do sistema que está sendo desenvolvido. O sistema de internacionalização do iVProg é bastante simples, refere-se apenas a mudança de idiomas. Não há necessidade de mudanças específicas relacionadas a padrões de datas ou valores monetários uma vez que o iVProg não utiliza esses conceitos.

Um importante aspecto da internacionalização é que o sistema não é reescrito para ser adaptado a outro idioma ou cultura. Todas as alterações são feitas em arquivos de configuração, que não exigem conhecimento de programação. No iVProg, a internacionalização que implementamos, por enquanto restringe-se ao idioma. Adaptações em aspectos como formato de data e formatação de unidades monetárias não foram implementadas. Os termos traduzidos para o português estão no arquivo

```
edu/cmu/cs/stage3/resourceBundle/i18n_pt_BR.properties
```

Neste arquivo, estão as chaves utilizadas pelo iVProg e seus respectivos valores, ou seja, as frases ou palavras em português.

Ao contrário do sistema base *Alice*, no iVProg é possível incorporar uma nova língua de modo simples. Para isso deve-se criar um novo arquivo similar ao `i18n_pt_BR.properties` sendo que `pt_BR` deve ser substituído por caracteres que indiquem a língua e o país. Dentro do arquivo, as chaves devem ser mantidas, e apenas os valores (ou seja, os termos) devem ser traduzidos.

A atual versão do iVProg conta com dois idiomas, português e inglês, respectivamente definidos em `i18n_pt_BR.properties` e `i18n_en_US.properties`.

Para executar o iVProg no modo aplicativo com uma língua diferente da portuguesa, definida como a língua padrão do sistema, é necessário fornecer o parâmetro `lang` conforme o exemplo a seguir que utiliza o inglês:

```
java -jar ivprog.jar lang=en
```

O conteúdo de lang pode ser "lang=pt" ou "lang=en".

3.2 Enriquecendo páginas Web com o iVProg

Uma página Web é um arquivo de texto que utiliza o formato *HTML* (HyperText Markup Language) e que deve ter a extensão *.html*. Este tipo de arquivo pode ser visualizado com o uso de navegadores Web como o Mozilla Firefox, Konqueror, dentre outros. Além disso, a página pode executar mini-aplicativos, mais conhecidos como *applets* (programas que podem ser executados pelo navegador).

O iVProg é facilmente integrável a páginas Web. A utilização se dá através da tag *APPLET* e os parâmetros necessários para uma chamada ao iVProg são mostrados no código da figura 3.2.

```
<APPLET CODEBASE="." ARCHIVE="ivprog.jar"
  CODE="edu.cmu.cs.stage3.alice.authoringtool.JAlice.class"
  WIDTH=791 HEIGHT=530>
  <param name="MA_PARAM_Proposition"
    value="%3Celement+class%3D%27edu.cmu.cs.stage3.alice.core.World%27
      %3D%27main%27%3E%3Cchild+filename%3D%27play%27%3E1_CDATA_INIT%3C
      element+class%3D%27edu.cmu.cs.stage3.alice.core.response.UserDef
      inedResponse%27+name%3D%27play%27%3E%3Cchild+filename%3D%27lista
      %27%3E2_CDATA_INIT%3Celement+class%3D%27edu.cmu.cs.stage3.alice.
      core.Variable%27+name%3D%27lista%27%3E%3Cchild+filename%3D%27__U
      nnamed0_%27%3E3_CDATA_INIT%3Celement+class%3D%27edu.cmu.cs.stag
      e3.alice.core.List%27%3E%3Cproperty+componentClass%3D%27java.lang
      g.Number%27+name%3D%27values%27%3E%3Citem+class%3D%27java.lang.D
      ouble%27%3E1.0%3C%2Fitem%3E%3Citem+class%3D%27java.lang.Double%2
      7%3E3.0%3C%2Fitem%3E%3Citem+class%3D%27java.lang.Double%27%3E12.
      0%3C%2Fitem%3E%3Citem+class%3D%27java.lang.Double%27%3E8.0%3C%2F
      item%3E%3C%2Fproperty%3E%3Cproperty+name%3D%27valueClass%27%3Eja
      va.lang.Number%3C%2Fproperty%3E%3Cproperty+name%3D%27data%27%3E%">
  </APPLET>
```

Figura 3.2: Código com os parâmetros necessários para a chamada do iVProg em uma página Web.

Este código deve ser acrescentado à página Web na qual deseja-se acrescentar o iVProg.

Na tag `ARCHIVE` deve se colocar o `.jar` do `iVProg`. Os valores de `WIDTH` e `HEIGHT` podem ser modificados de acordo com a largura e altura da *applet* na página *HTML*.

Através do parâmetro `MA_PARAM Proposition` é possível enviar para um servidor o conteúdo de uma sessão, ou seja, tudo que foi feito pelo usuário no `iVProg`. O `value` deste parâmetro é codificado pelo `iVProg` apenas para evitar erros de `encode` durante a comunicação entre o servidor e o `iVProg`.

3.2.1 Gerando uma página Web com o iVProg

O `iVProg` possui a funcionalidade de criação automática de uma página *HTML* através do menu *Exercício* e em seguida o submenu *Veja código HTML* como ilustrado na figura 3.3. Ao acessar esse menu, será exibida uma janela com o código *html* do programa/álgoritmo da sessão, ou seja, o algoritmo construído no `iVProg`.

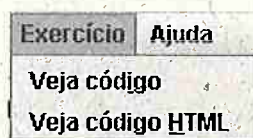


Figura 3.3: Funcionalidade criação de páginas Web com o `iVProg`.

Assim, o professor pode gerar páginas *HTML* que apenas mostrem o trabalho realizado em uma sessão de uso do `iVProg` (para usar como exemplo em uma aula) ou páginas contendo exercícios, que poderão ser publicadas em um servidor web para que seus alunos acessem pela internet e resolvam os exercícios.

Para criar tais páginas, não é necessário qualquer tipo de configuração. A grande vantagem oferecida pelo programa é que qualquer usuário pode gerar estas páginas sem ter o menor conhecimento de programação. Para entender esta funcionalidade, suponhamos o seguinte exemplo: deseja-se construir uma página Web com o programa/álgoritmo constante na figura 3.4. Este programa recebe dois números, em seguida determina e mostra a média aritmética.

Através dessa funcionalidade, o sistema mostra uma janela com o código *html*. E esse código criado conterá o `iVProg` com o referido programa/álgoritmo como ilustra a figura 3.5

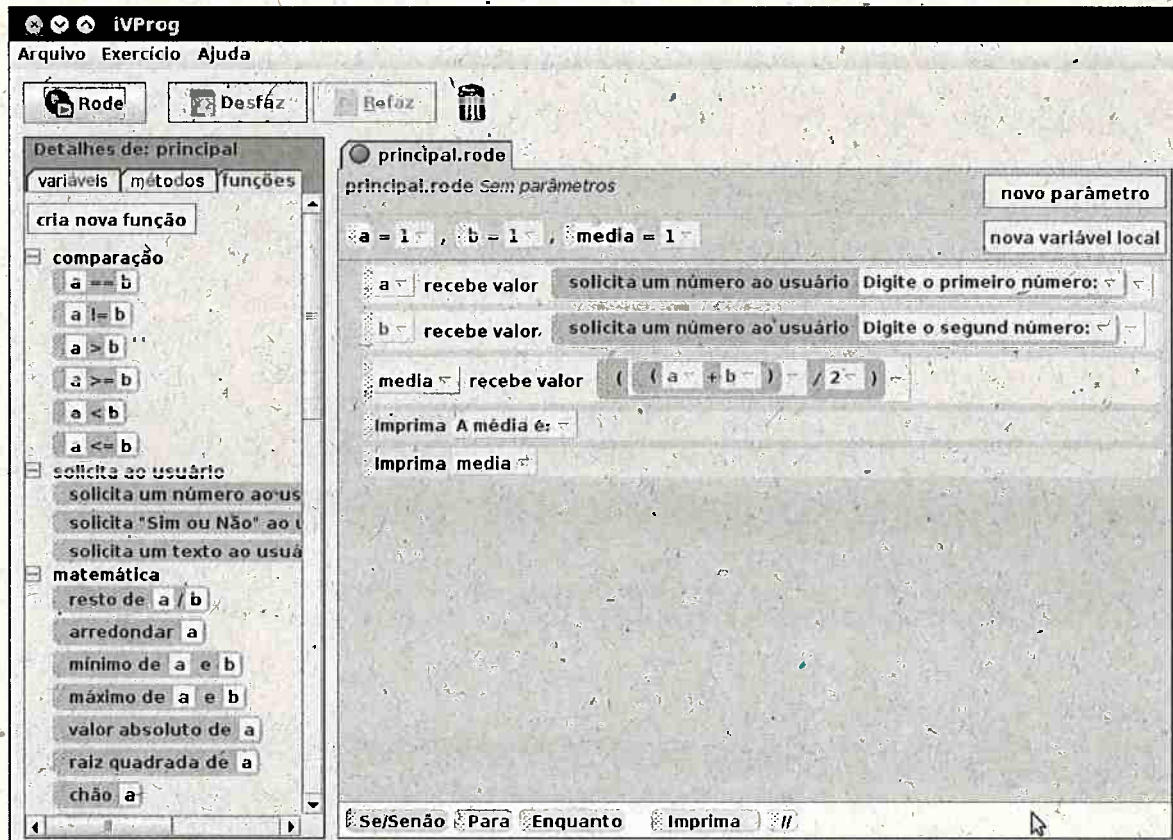


Figura 3.4: Exemplo de programa/algorithmo no iVProg.

3.2.2 iVProg como Widget

Widgets são componentes computacionais integráveis a páginas Web através da simples operação de copiar e colar pedaços de código *HTML*. A simplicidade e independência dos widgets permite que qualquer pessoa possa integrá-los a blogs ou páginas pessoais. A figura 3.6 mostra a disponibilidade da widget do iVProg.

A disponibilização do iVProg como widget, de maneira a possibilitar sua incorporação em blogs ou páginas pessoais tem o potencial de aumentar e estimular conteúdos relativos a introdução à programação pelo mundo virtual. A atual implementação permite a inclusão da *widget* do iVProg através da simples operação de copiar e colar, bastando a cópia do trecho específico para a página Web.

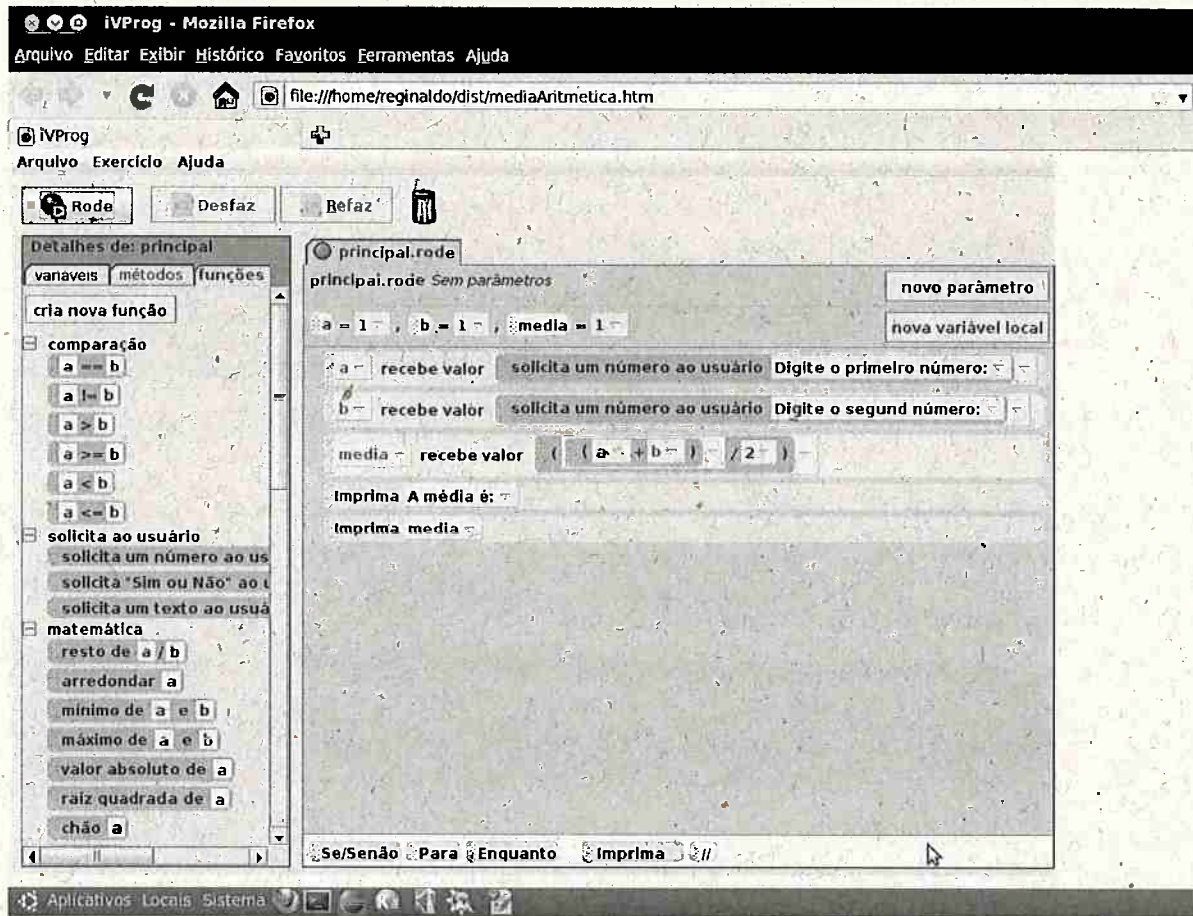


Figura 3.5: Página Web com o programa gerado na figura 3.4.

3.3 iVProg como iMA

Um iMA é um programa na forma de *applet*⁵, que disponha de recursos para comunicação com um servidor, tanto para receber conteúdos quanto para enviá-los (RODRIGUES & BRANDÃO, 2009). Deste modo, são necessários apenas 3 novos métodos para que um *applet* possa ser considerado um iMA. Do ponto de vista do servidor, pode-se implementar métodos para apresentar o iMA dentro do navegador e para receber os conteúdos desenvolvidos por alunos no iMA.

Assim, é possível disponibilizar para os alunos exercícios elaborados no iMA e os alunos podem, com um *clique*, enviar suas soluções ao servidor.

Se o iMA possui avaliador automático, o professor pode ter sua tarefa reduzida, por ser

⁵ Um *applet* é um mini-aplicativo *Java* (<http://java.sun.com>) que pode ser utilizado dentro de qualquer navegador.



```

<applet codebase="." archive="ivprog.jar"
code="edu.cmu.cs.stage3.alice.authoringtool.JAlice.class" WIDTH=791 HEIGHT=530>
  <param name='MA_PARAM_Proposition' value='<element
class="edu.cmu.cs.stage3.alice.core.World" name="principal"><child
filename="rode">1 CDATA_INIT<element
class="edu.cmu.cs.stage3.alice.core.response.UserDefinedResponse" name="rode"><property
name="requiredFormalParameters"></property><property
name="keywordFormalParameters"></property><property
name="localVariables"></property><property name="componentResponses"></property><property
class="java.lang.Double" name="duration">1.0</property><property
name="data"></property></element>1 CDATA_END</child><child
filename="rode">1 CDATA_INIT<element
class="edu.cmu.cs.stage3.alice.core.response.UserDefinedResponse" name="rode"><property
name="requiredFormalParameters"></property><property
name="keywordFormalParameters"></property><property
name="localVariables"></property><property name="componentResponses"></property><property
class="java.lang.Double" name="duration">1.0</property><property
name="data"></property></element>1 CDATA_END</child><child
filename="behavior0">1 CDATA_INIT<element
class="edu.cmu.cs.stage3.alice.core.behavior.WorldStartBehavior" name="behavior0"><child
filename=" _Unnamed0_ ">2 CDATA_INIT<element
class="edu.cmu.cs.stage3.alice.core.response.CallToUserDefinedResponse"><property
criterionClass="edu.cmu.cs.stage3.alice.core.criterion.InternalReferenceKeyedCriterion"
name="userDefinedResponse">rode</property><property
name="requiredActualParameters"></property><property
name="keywordActualParameters"></property><property class="java.lang.Double"
name="duration">1.0</property><property
name="data"></property></element>2 CDATA_END</child><child
filename=" _Unnamed1_ ">2 CDATA_INIT<element
class="edu.cmu.cs.stage3.alice.core.response.CallToUserDefinedResponse"><property
criterionClass="edu.cmu.cs.stage3.alice.core.criterion.InternalReferenceKeyedCriterion"

```

Figura 3.6: Widget do iVProg.

dispensado de avaliar cada um dos exercícios, e ainda dispor de facilidades como o acesso instantâneo ao desempenho de seus alunos - que permite identificar eventuais dificuldades em determinados exercícios (RODRIGUES & BRANDÃO, 2009). Já para o aluno, a grande vantagem é saber imediatamente se sua resposta está de acordo com o esperado, evitando-se assim uma das grandes fontes de desmotivação em cursos na Web, que é a demora em ter sua atividade avaliada (HARA & KLING, 1999; HENTEA et al., 2003). Esse recurso de avaliação automática não foi implementada no iVProg até o momento, mas é um trabalho futuro que será priorizado.

O iVProg, que foi implementado como iMA é facilmente integrável com um Sistema Gerenciador de Cursos (SGC), como por exemplo o SAW (MOURA, 2007) ou Moodle (RODRIGUES & BRANDÃO, 2009). A comunicação entre servidor e o iVProg se dá através de parâmetros do *applet*, em outras palavras, o cliente solicita uma requisição para o servidor e o mesmo responde com uma página *HTML* e a sua respectiva *tag APPLET* devidamente montada.

Uma vez concluído o exercício as informações do iVProg devem ser enviadas ao servidor. Esta comunicação, na direção iVProg para o servidor é realizada por um método *JavaScript*

específico que irá requisitar do *applet* as informações e enviá-las ao servidor a partir de um método tipo *POST*⁶.

A figura 3.7 apresenta três etapas do ciclo de vida da utilização do *iVProg* como *iMA*.

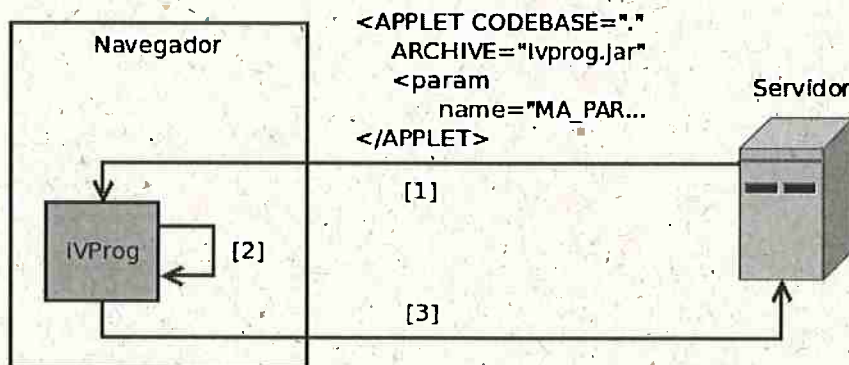


Figura 3.7: *iVProg* como *iMA*.

A etapa [1] corresponde ao recebimento do conteúdo para apresentá-lo ao aluno. Isso é feito a partir de *tags* do *applet*, por exemplo, usando

`MA_PARAM_adresPOST` com o endereço no servidor que tratará a resposta do aluno;

`MA_PARAM_Proposition` usando como valor o conteúdo a ser mostrado ao aluno.

Nesta etapa, o servidor no qual se encontra o *SGC* deve estar preparado para tratar a resposta a ser enviada pelo aluno.

A etapa [2] representa a execução do *iVProg*. É importante observar que durante a operação do *iVProg* o servidor não é mais solicitado e todo o processamento ocorre no navegador cliente.

A etapa [3] representa o envio da resposta fornecida pelo aluno ao servidor. O código *javascript* a seguir ilustra a comunicação no sentido *iVProg*-servidor. O método `getResposta()` é capaz de acessar diretamente o método público `getAnswer()` da *applet*. Esse método retorna a resposta do aluno, na forma de programa/álgebra, para enviá-los ao servidor.

```
<script type="text/javascript">
  function getResposta() {
    var resposta = document.ivprog.getAnswer();
    ...
  }
```

⁶ Método para envio de dados a partir de formulários, para mais informações pode-se examinar o sítio *W3C* <http://www.w3.org/>.

```
}  
</script>
```

Este método pode ser útil para atividades em que existam mais de um exercício por página Web do SGC. O professor poderá criar, por exemplo, um trabalho no qual o aluno deverá enviar o programa/ algoritmo, responder questões dissertativas e de múltipla escolha na mesma página. O exemplo a seguir mostra que é possível colocar 3 “instâncias” do iVProg em uma página *html* e submeter as respostas ao servidor através do método `submit_MA_Answer()`.

```
<script Language="JavaScript">  
function submit_MA_Answer () {  
    var resposta_exerc = new Array(3);  
    var sessao = new Array(3);  
    var doc = document.eMA_0;  
    resposta_exerc[0] = doc.getAnswer();  
    var doc = document.eMA_1;  
    resposta_exerc[1] = doc.getAnswer();  
    var doc = document.eMA_2;  
    resposta_exerc[2] = doc.getAnswer();  
    docForm.MA_POST_Archive_0.value = resposta_exerc[0];  
    docForm.MA_POST_Archive_1.value = resposta_exerc[1];  
    docForm.MA_POST_Archive_2.value = resposta_exerc[2];  
    docForm.submit();  
    return true;  
}  
</script>
```

Por outro lado, caso exista uma simples atividade em que exista apenas um exercício por página, é possível utilizar o método público `sendPOST()` da *applet* que enviará a resposta diretamente ao SGC.

Capítulo 4

Experimentos

Já foram realizados dois experimentos com o iVProg, ambos com alunos do curso de Licenciatura em Matemática no Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP), em uma disciplina de introdução à computação, no primeiro semestre de 2010. Denominaremos esta turma por T1.

O primeiro foi um experimento controlado, em laboratório e com observadores, já o segundo foi o uso do iVProg para introduzir programação no semestre regular.

4.1 Exercícios observados

O primeiro experimento foi constituído por 2 exercícios, com os alunos da T1 divididos aleatoriamente em 2 grupos, G1 e G2, em aproximadamente 1 hora. O grupo G1 deveria fazer ambos os exercícios primeiro usando o iVProg e depois usando um ambiente tradicional em C. Já G2 resolveria os mesmos exercícios primeiro em C e depois em iVProg. Cada aluno deveria resolver individualmente seus exercícios sem consulta, tendo um observador ao seu lado para tomar nota, por exemplo, de quantos erros sintáticos ou quantas vezes o programa foi executado até o aluno considerá-lo como correto.

Este experimento teve como objetivo analisar as hipóteses iniciais de que o iVProg simplifica a tarefa de programação para alunos iniciantes, além de avaliar a usabilidade do sistema. A usabilidade foi baseada em um conjunto de heurísticas desenvolvidas pelo engenheiro dinamarquês Jakob Nielsen (NIELSEN, 2000), que identificou que experimentos com apenas 5 usuários seriam suficientes para identificar 85% dos problemas de usabilidade de um software.

Todos os alunos de T1 foram convidados para participarem do experimento, que foi realizado durante o horário semanal da aula de monitoria a turma. No horário combinado apareceram apenas 6 alunos. Deve-se notar que as médias finais do alunos de ambos os grupos ficaram bem próximas (G1 4,98 com desvio padrão de 3,19 e G2 com 4,39 e desvio padrão de 1,73). É importante destacar que a maioria destes alunos não possui conhecimento anterior na área de programação.

Inicialmente foi explicado o experimento, número de exercícios e o papel dos observadores. Uma vez dividida a turma, foi distribuído um questionário com apenas 2 perguntas, visando registrar a experiência dos alunos com a programação:

Q1) Possui conhecimento da linguagem C? Qual experiência?

Q2) Já fez a disciplina de Introdução à Computação? Utilizou qual linguagem/ferramenta?

Recolhidos os questionários, liberamos os exercícios para os grupos. Eles deveriam resolver os exercícios e enviar suas soluções para o SGC. Os exercícios foram:

- 1. Fazer um programa que solicite dois números e mostre sua média aritmética.*
- 2. Fazer um programa que solicite um número natural n e uma sequência de n números. Em seguida o programa deve imprimir na tela o maior e o menor número.*

Cada observador deveria anotar ao menos os seguintes itens:

Para os exercícios quando feitos usando iVProg

- a) tempo para resolver o exercício
- b) número de vezes que rodou o programa
- c) número de vezes que detectou erros de sintaxe ou lógica

Para os exercícios quando feitos em C

- A) tempo para resolver o exercício
- B) número de vezes que compilou
- C) número de vezes que rodou
- D) número de vezes que detectou erros de sintaxe ou lógica

O objetivo dos exercícios e seus itens observados é identificar as dificuldades dos alunos com a linguagem e/ou ferramenta. Por esse motivo, foram escolhidos os exercícios que

envolvem uma lógica simples. Outro fator motivador para esta escolha, foi o tempo limite para a realização do experimento, que foi de apenas 1 hora com o objetivo de identificar as dificuldades iniciais dos alunos nestas linguagens/ferramentas.

Após a aplicação do experimento foi feita a digitalização dos dados e, em seguida, organizados na tabela 4.1.

Tabela 4.1: Comparativo entre iVProg e a linguagem C nos critérios observados.

		GRUPO 1				GRUPO 2				GERAL
		aluno1	aluno2	aluno3	média	aluno4	aluno5	aluno6	média	média
	Q1	não	não	não	-	sim	não	não	-	-
	Q2	sim	sim	não	-	sim	sim	não	-	-
Ex. 1 no iVProg	a)	2	1	3	2,0	2	5	2	3	2,5
	b)	1	2	3	2,0	1	1	1	1	1,5
	c)	0	0	0	0,0	0	0	0	0	0,0
Ex. 2 no iVProg	a)	8	2	14	8,0	13			13	10,5
	b)	1	1	2	1,3	5	não concluiu	não concluiu	5	3,2
	c)	0	0	0	0,0	3			3	1,5
Ex. 1 em C	A)	9	5	15	9,7	23	50	35	36,0	22,8
	B)	2	1	5	2,7	4	10	11	8,3	5,5
	C)	2	1	2	1,7	2	1	11	4,7	3,2
	D)	1	0	1	0,7	3	6	10	6,3	3,5
Ex. 2 em C	A)	16	5	21	14,0	30			30	22,0
	B)	4	5	5	4,7	5	não concluiu	não concluiu	5	4,8
	C)	4	1	2	2,3	4			4	3,2
	D)	4	5	2	3,7	4			4	3,8

Na tabela 4.1, a linha "Q1" aponta o primeiro item do questionário, ou seja, se o aluno possui experiência com a linguagem C. A linha "Q2" é o segundo item do questionário, que indica se o aluno já fez a disciplina de Introdução à Computação. As linhas "a)", "b)" e "c)" indicam exercícios feitos no iVProg e seus respectivos itens observados: tempo para resolver o exercício; número de vezes que rodou o programa e número de vezes que detectou erros de sintaxe ou lógica. Já as linhas "A)", "B)", "C)" e "D)" indicam exercícios feitos C e seus respectivos itens observados: tempo para resolver o exercício; número de vezes que compilou; número de vezes que rodou e número de vezes que detectou erros de sintaxe ou lógica.

A partir das respostas do questionário, notamos que nenhum dos alunos de G1 possui conhecimento ou experiência na área de programação, enquanto que em G2 apenas 1 possui. Portanto, a maioria dos alunos que participaram do experimento se enquadram no perfil do público alvo deste projeto.

Analisando-se os resultados fica clara a vantagem de utilizar o iVProg. Observa-se, por

exemplo, que todos os alunos de G1 resolveram com sucesso os 2 exercícios, enquanto apenas 1 de G2 resolveu ambos. A partir das observações ficou claro que todos os alunos G2 tiveram grande dificuldade com a sintaxe da linguagem C, com vários erros sintáticos e lógicos, por exemplo, os alunos 5 e 6 precisaram compilar mais de 10 vezes para que primeiro exercícios ficasse correto. O aluno 4 apresentou menor número de compilações, mas também foi observada grande dificuldade com a sintaxe para conseguir escrever um código correto.

Outro fato que demonstra maior facilidade dos alunos com o iVProg é o tempo médio para resolver os exercícios. Com o iVProg o tempo sempre foi menor, mesmo considerando o grupo 1 que primeiro deduziu os algoritmos com o iVProg e apenas os implementaram em C: o tempo médio com o iVProg foram 2 min e 8 min, contra 9,7 min e 14 min dos mesmos exercícios em C.

4.2 Uso em disciplina regular

Este experimento corresponde ao uso do iVProg para introduzir programação com a turma T1, mas neste item analisaremos apenas a usabilidade do iVProg. Questões de aprendizagem serão discutidas nos próximos itens. Ao final do semestre, 32 alunos (de 48) desta disciplina responderam a um questionário, composto por 7 perguntas. As questões eram de múltipla escolha com o objetivo de reduzir a subjetividade das opiniões e facilitar a tabulação dos dados. Para fazer o levantamento das preferências dos participantes foi utilizada a escala proposta por Renis Likert (apud OLIVEIRA, 2005), que consiste em disponibilizar cinco conceitos de múltipla escolha que variam conforme o seguinte exemplo de Likert:

- Péssimo
- Ruim
- Regular
- Bom
- Ótimo

Nas próximas páginas, seguem o enunciado, o objetivo de cada questão, o respectivo gráfico (baseado na contagem do número de ocorrências de cada alternativa) e uma breve análise dos resultados.

Questão 1) Quantas horas aproximadamente você utiliza o computador/internet por dia? (independente da finalidade)

- a) Menos que 1 hora
- b) Entre 1 e 2 horas
- c) Entre 2 e 4 horas
- d) Entre 4 e 6 horas
- e) Acima de 6 horas

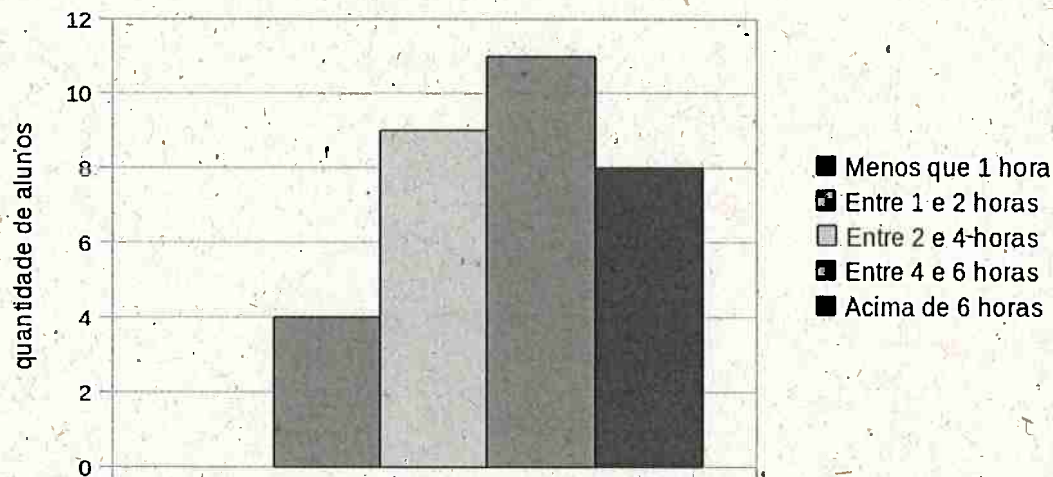


Figura 4.1: Frequência de uso do computador.

Objetivo: O objetivo desta questão foi identificar o grau de afinidade do aluno com o computador, possibilitando a identificação de eventuais dificuldades com o uso do programa decorrentes de inexperiência com o uso da máquina.

Análise do resultado: Nota-se que a maioria dos alunos são usuários frequentes de computador e, em decorrência desse fato, que as dificuldades eventualmente apresentadas por essas pessoas, provavelmente, não estão ligadas à inexperiência com o uso do equipamento.

Questão 2) Considerando o aspecto visual, o iVProg lhe parece um programa:

- a) Incompreensível
- b) Difícil compreensão
- c) Indiferente
- d) Compreensível
- e) Muito simples

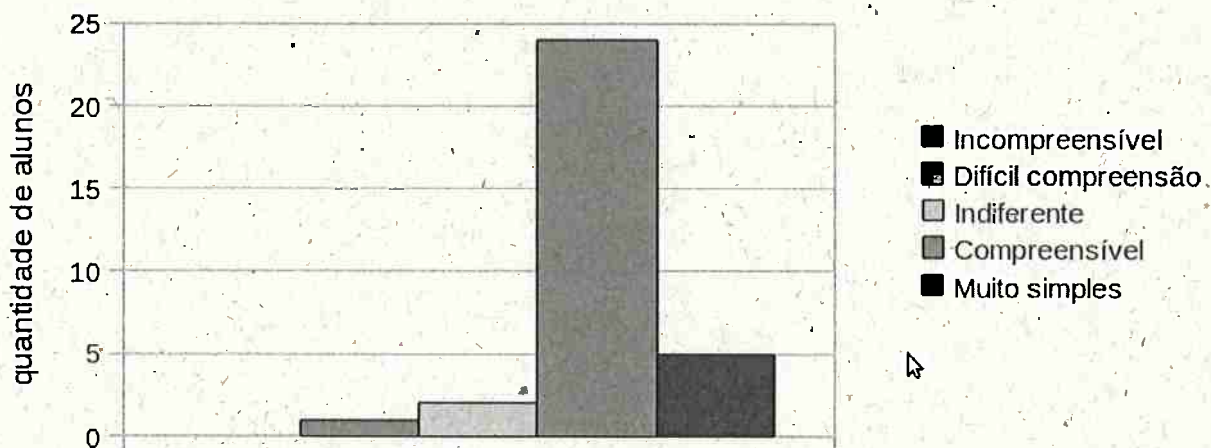


Figura 4.2: Opinião dos alunos com relação ao aspecto visual do iVProg.

Objetivo: O objetivo desta questão é verificar se a interface do iVProg é de fácil compreensão.

Análise do resultado: A partir das respostas, nota-se uma clara aprovação do iVProg, sendo que 75% dos alunos responderam que a sistema é compreensível.

O gráfico mostra que houveram poucas opiniões negativas sobre o aspecto visual. A observação direta do gráfico permite concluir que o aspecto visual do programa foi aprovado pelos alunos que responderam o questionário.

Questão 3) Na sua opinião, para quem tem um primeiro contato, o iVProg é:

- a) Muito difícil
- b) Difícil
- c) Simples
- d) Fácil
- e) Muito Fácil

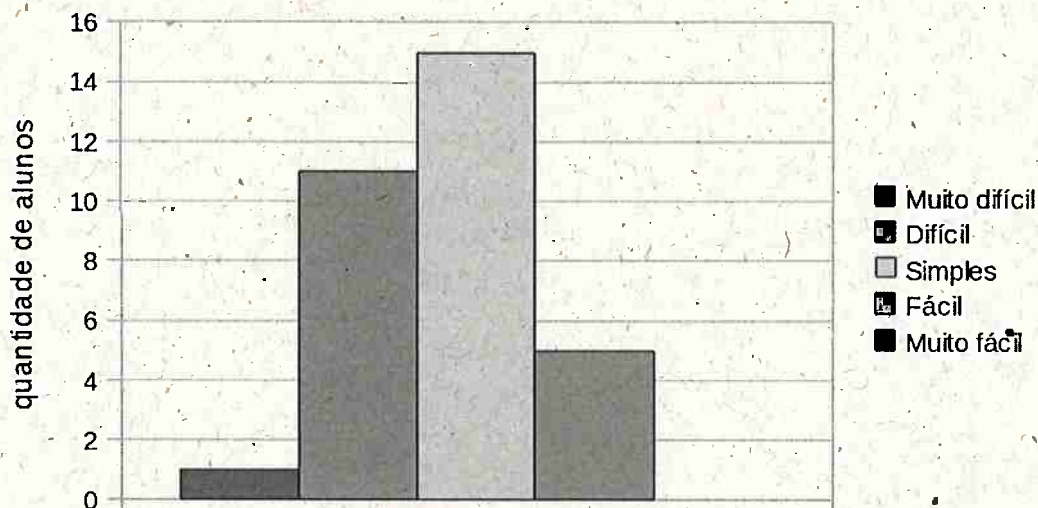


Figura 4.3: Opinião dos alunos em relação ao primeiro contato com o iVProg.

Objetivo: Descobrir o quão confortáveis se sentem os alunos ao usar o programa pela primeira vez.

Análise do resultado: A maioria das pessoas considerou o uso do programa de simples a fácil já na primeira experiência. Alguns alunos, no entanto, deram opiniões negativas sobre a facilidade no uso inicial. Isto aponta que melhorias devem ser feitas para tornar mais simples o primeiro contato do usuário com o programa.

Questão 4) De modo geral, resolver exercícios utilizando o iVProg é:

- a) Muito desagradável
- b) Desagradável
- c) Médio
- d) Agradável
- e) Muito agradável

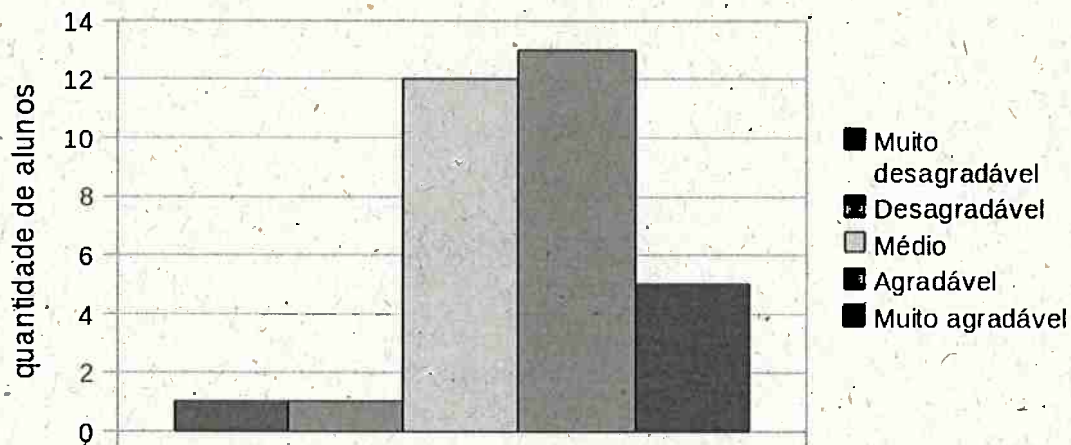


Figura 4.4: Opinião dos alunos com relação à resolução de exercícios no iVProg.

Objetivo: Descobrir o quão confortáveis se sentem os alunos ao usar o programa pela primeira vez.

Análise do resultado: A resolução de exercícios no iVProg foi considerada agradável por 40,6% dos participantes. No entanto, existiu a reprovação de 6,2% das pessoas, por acreditarem ser desagradável ou muito desagradável. Mais uma vez, a maioria das opiniões foi favorável, indicando que, embora haja correções a fazer, a implementação atual é agradável o suficiente para a maior parte dos alunos.

Além das questões objetivas, o questionário contou ainda com três questões discursivas. Seguem abaixo o enunciado dessas questões e algumas frases escritas pelos alunos. Nas frases transcritas foi mantida a grafia original.

Questão 5) Por favor, destaque as principais vantagens de programar com o iVProg em relação à linguagem C:

Esta era uma questão discursiva e destacamos algumas das respostas mais frequentes. As opiniões a seguir reforçam que o iVProg evita erros na digitação de comandos e reduz os erros relacionados com sintaxe.

“Não tem risco de errar na digitação. Basta só pensar em algoritmo.”

“No iVProg, nos preocupamos menos com a sintaxe em si, dando mais atenção para a lógica da resolução.”

“Todos os comandos que em C devem ser digitados, no iVprog basta que arrastemos o ícone. O que torna mais hábil para manuseio.”

“Eu acho que a principal vantagem de usar o iVprog foi não precisar ficar preocupado com a sintaxe dos comandos. Isso com certeza nos ajuda a dar atenção ao algoritmo em questão.”

Além disso, alguns alunos destacaram a facilidade da visualização prévia dos comandos existentes:

“Com o iVProg tenho opções para ir tentando fazer o exercício enquanto em C, senão tiver noção de como começar, fica difícil partir do zero.”

“Visualização dos algoritmos, biblioteca de funções de fácil acesso previamente definidas”

Também foram relatadas vantagens da identificação utilizada na programação visual do iVProg:

“Os blocos de comandos ficam bem visíveis e fácil de entender.”

“Deixa claro o início e o fim de um laço pelo fato de utilizar ‘caixas’. Não exige que o usuário decore funções de comando, já que todas as ferramentas e funções estão apresentadas na interface do programa.”

E finalmente, alguns alunos destacaram a importância de facilidades no iVProg como inicialização e alteração de nome de variáveis além da internacionalização do sistema:

“Sintaxe pronta, controle de variável, alteração no nome da variável altera em todo o programa.”

“A não necessidade de ‘decorar’ comandos - A organização em blocos oferecida pelo iVprog - Atratividade visual - Comandos em Português”

Os depoimentos dessa questão confirmaram que de fato o iVProg é mais prático e adequado aos alunos iniciantes por utilizar o modelo da programação visual.

Questão 6) Por favor, destaque as principais vantagens de programar usando a linguagem C em relação ao iVProg:

Esta também foi uma questão discursiva e foram destacadas as respostas mais frequentes. Alguns alunos declararam não existir vantagens de programar usando a linguagem C em relação ao iVProg:

“Não enxergo vantagens porque creio que o iVProg tem uma compreensão um pouco mais fácil.”

“Desconheço.”

Entretanto, a maioria dos depoimentos indicam que o iVProg é uma ferramenta didática, não para produzir programas profissionais de grande porte.

“Se habituar com a linguagem de programação.”

“Desconheço, talvez a velocidade para compilar alguns programas mais sofisticados.”

“Em C, podemos utilizar compiladores ou editores de maneira mais flexível. Já o iVprog na versão web restringe um pouco seu uso, uma vez que seu bom funcionamento depende de fatores tais quais a qualidade da conexão, do próprio computador, etc. A versão aplicativa é mais flexível.”

“As vantagens são que você estará usando uma linguagem em que é possível fazer programas, e não só entendendo algoritmos. Assim você começa a entender mais o computador, seu funcionamento, as bibliotecas padrão etc. ”

“Rapidez em rodar um programa maior; *poder copiar e colar algum programa pronto, útil para programas muito longos; *melhor para compilar.”

Esses depoimentos indicam que boa parte dos alunos também tiveram a percepção de que o iVProg é uma ferramenta didática, não para produzir sofisticados.

Questão 7) Existe algum recurso que gostaria de ver implementado no iVProg? (em caso afirmativo favor citar qual)

Esta foi a última questão discursiva. Foram destacadas as sugestões mais frequentes e a seguir são apresentadas as que sugerem a criação de um arquivo executável:

“Sim. Seria interessante que os programas feitos com o iVProg gerassem um arquivo executável.”

“Poder abrir o programa feito pela iVprog diretamente pelo programa, e não por meio do iVprog. Como em C com .exe.”

Já os depoimentos abaixo sugerem a criação de mensagens de ajuda e ferramentas para o reaproveitamento de código:

“Sim. um recurso que apontasse ou desse uma dica do erro cometido”

“Criação de uma área comum para armazenamento de funções para reaproveitamento em outro código. O padrão de sempre iniciar sem nada não facilita a reutilização objetos.”

A seguir, as sugestões que indicam a existência de pequenos pontos que devem ser melhorados no iVProg:

“Acho que o iVProg já é um programa bom e não necessita de implementações, porem acho que o curso deveria escolher ou usar o iVProg, ou usar linguagem C, pois a transição de um para outro se torna muito mais difícil.”

“Quando estávamos no aplicativo, o programa nos alertava de tempo em tempo para salvar o que foi feito, já online isso não acontece e se o programa travar ou acontecer qualquer outro imprevisto perdemos tudo. Talvez se houvesse um jeito de salvar online seria bom!”

“Os textos no iVprog [prints] não estão muito legais, pois você não pode fazer adequações para o programa ficar bem visto, exemplo, : ”O Numero 'x' esta na coluna 'y' ”, etc”

“Independência dos laços de if/else, já que, nem sempre é necessário utilizarmos o else na função if.”

“Talvez o principal problema encontrado por todos no iVprog, foi o travamento exagerado quando tentávamos rodar o programa. E como algumas vezes esse erro se dava a alguma janela aberta, acredito que poderia ser impresso na tela um aviso indicando que a janela ainda estava aberta. Também acredito que se os exercícios pudessem ser salvos quando o fazemos de modo online facilitaria muito, pois como eles não podiam ser entregues em arquivos compactados, tínhamos que fazê-los na internet, e muitas vezes perdíamos o que

havíamos feito por causa de algum problema que o iVprog apresentava.”

Estas foram boas sugestões de usabilidade, que devem ser consideradas em trabalhos futuros.

4.2.1 Comparativo de desempenho com turmas anteriores

A fim de realizar uma análise comparativa do uso do iVProg, os resultados da turma T1 foram comparados com dados de outra turma da mesma disciplina e curso, ministrada em 2005 pelo orientador dessa dissertação. Esta segunda turma, que será designada T2, teve 50 alunos efetivos (alunos com ao menos uma atividade valendo nota), enquanto T1 teve 48. Em ambas as turmas, todas as aulas ocorreram em laboratório, com no máximo 2 alunos por computador. O método de apresentação era propor um exercício/problema à turma e incentivar os alunos a apresentarem sugestões.

Os principais tópicos cobertos na disciplina foram: Média de n números; Algoritmo de Arquimedes para MDC; Computar n ésimo termo de Fibonacci; Testes com tipo, problemas com 'char' e 'int'; Computar $\cos(x)$ via aproximação por série de Taylor; Vetores: ordena um vetor utilizando uma função auxiliar (para achar um mínimo); Vetores: busca binária; Vetores: representação de conjuntos e Função que verifica-se um vetor é de permutação.

Método de ensino utilizado na turma T2

Na turma T2, introduziu-se o conceito de programação a partir de um sistema que simulava o computador, utilizando-se inicialmente uma programação na linguagem desta máquina simulada. Comandos *alto-nível*, em linguagem C, foram deduzidos a partir do modelo *baixo-nível*. O emulador também dispunha de um compilador para sua linguagem de máquina e pode ser livremente descarregado pela Web (<http://www.matematica.br/programas/icg/>). A partir da semana 5 passou-se a utilizar apenas a linguagem C.

A avaliação foi composta por 2 provas, 5 exercícios/programas *online* e 3 exercícios/programas (exercícios maiores envolvendo também a modelagem de problema).

Método de ensino utilizado na turma T1

Nesta turma iniciou-se o conteúdo diretamente em uma linguagem *alto-nível* usando o iVProg. Na medida em que o curso avançava, apresentava-se os comandos numa sintaxe do tipo C. A avaliação foi composta por 2 provas, sendo a primeira online (um aluno por computador) e a segunda teórica/tradicional, foram feitos 33 exercícios/programas *online*,

no iVProg, e 20 exercícios/programas feitos em C.

Análise comparativa das turmas

Com o objetivo de fazer uma análise comparativa entre T1 (turma que utilizou o iVProg) e T2 (turma que não utilizou o iVProg), foi feita a tabulação dos dados que podem indicar o desempenho no aprendizado durante o semestre, como por exemplo: notas nas avaliações, exercícios feitos, presença e finalmente a nota média final. Estes dados são apresentados na tabela 4.2.

Tabela 4.2: Comparativo das 2 turmas

	M T1	DP T1	M T2	DP T2
presença	73,37%	0,268	56,48%	0,330
avaliação 1	5,49	2,98	3,28	2,88
avaliação 2	3,58	2,20	4,33	4,12
exercícios <i>online</i>	9,32	1,61	7,00	4,52
exercícios/programas	8,57	2,28	4,28	4,02
nota final	4,64	2,27	4,06	3,23

A coluna “M T1” indica a média na turma T1, enquanto “DP T1” indica o seu desvio padrão. A coluna “M T2” indica a média na turma T2, enquanto “DP T2” indica o seu desvio padrão. A linha “presença” indica a presença média dos alunos durante o semestre. As linhas “avaliação 1 e 2” representam as notas obtidas pelos alunos nestas avaliações. A linha “exercícios/programas *online*” indica notas obtidas nesta modalidade composta por exercícios com baixo grau de dificuldade. A linha “exercícios/programas” indica notas obtidas nos exercícios maiores envolvendo também a modelagem de problema.

Uma variável importante na análise de desempenho das turmas é a presença em aula, pois pode indicar maior ou menor motivação do aluno com a disciplina. Neste sentido, o comparativo indica clara vantagem do iVProg, pois enquanto na turma 1 a presença média foi 73,37% (com desvio padrão de 0,268), na turma 2 a presença média em aulas foi de apenas 56,48% (desvio padrão um pouco maior, 0,330). O maior desvio padrão em T2 pode ser explicada pelo maior número de desistentes (14 desistentes em T2 contra apenas 7 em T1).

Com relação às notas, a média da primeira avaliação em T1 (5,49) é maior que em T2 (3,28), indicando maior eficácia do uso do iVProg, pois como já citado, na primeira avaliação de T1, todos optaram por fazê-la no iVProg.

Na turma T1, cabe um destaque para a análise comparativa entre iVProg e a linguagem C. Na primeira avaliação os alunos podiam optar por fazer em C ou no iVProg. Dos 46 alunos

que fizeram a avaliação, todos optaram pelo iVProg. Porém, aqui deve-se apontar um viés pró-iVProg que foi o número de semanas com cada uma das linguagens: 6 semanas exclusiva com iVProg contra 2 usando apenas C.

Tabela 4.3: Comparativo dos exercícios feitos em T1.

	nota		quantidade entregue	
	média	desvio padrão	média	desvio padrão
exercícios feitos no iVProg	9,32	1,61	69,1%	0,462
exercícios feitos em C	8,57	2,28	45,5%	0,497

Outros indicadores da eficácia do iVProg foram : a brusca queda de envio de soluções para os exercícios, quando estes passaram a ser cobrados apenas em C a média de envios caiu de 69,1% para 45,5% (tabela 4.3); e o resultado da avaliação 2, cuja média foi 5,49 na avaliação 1 contra 3,58 na avaliação 2. Uma possível razão para a queda drástica de desempenho nas avaliações em T1 foi precisamente a queda no número de exercícios feitos pelos alunos. Durante as aulas, quando indagados sobre a queda no envio de exercícios, a justificativa dos alunos foi maior dificuldade com a linguagem C.

Nas avaliações feitas em C, observou-se, nas duas turmas, um elevado número de erros de sintaxe, como o erro em expressão lógica: usar "a=1" no lugar de "a==1". Esse é um típico erro semântico em linguagem C, que gerou frustração para alguns alunos, sendo sempre necessária a intervenção do professor para apontar o erro.

4.3 Conclusões do capítulo

Nesse capítulo foram apresentados vários experimentos e métodos para avaliar as características do iVProg. Foi apresentado ainda o questionário utilizado como estratégia para a coleta de dados realizada no final do curso de *Introdução à Computação*. Após análise desses dados que indicou a existência de pontos que devem ser melhorados no iVProg, entretanto o programa foi bem aceito pelos alunos desta disciplina.

Capítulo 5

Conclusões

5.1 Considerações finais

Nas últimas décadas, a internet passou a ser utilizada como ferramenta importante no processo de ensino, sendo o incremento nos últimos anos particularmente empurrado pelo surgimento de sistemas gerenciadores de cursos (SGC). Esses sistemas propiciaram o acompanhamento remoto, por parte dos professores, de atividades desenvolvidas pelos alunos, facilitando ao aluno seguir seu próprio ritmo de aprendizagem.

O objetivo deste trabalho foi apresentar um sistema Web para introdução à programação que possa efetivamente ser utilizado em aulas como ferramenta de auxílio ao professor e ao aluno: o iVProg. A meta principal deste sistema é motivar os estudantes a assimilarem melhor o conceito de algoritmo, reduzindo problemas típicos dos ambientes tradicionais para ensino-aprendizagem de programação. Além disso, o iVProg evita boa parte dos erros de sintaxe muito comuns na programação (em disciplinas introdutórias).

Para verificar a validade desta proposta, o iVProg foi testado em aulas de introdução à programação. Os resultados apresentados através dos testes realizados mostraram que as diretrizes seguidas no sistema funcionaram de forma satisfatória. Vale destacar que este sistema funciona plenamente via Web e pode ser integrado a SGC. Esta integração foi utilizada com sucesso na disciplina aqui relatada (turma T1).

Como trabalhos a curto prazo no iVProg, destacamos a necessidade de finalizar a internacionalização. Mais a longo prazo, outra funcionalidade a ser desenvolvida é a avaliação automática a partir de um gabarito. Futuramente também esperamos implementar um ex-

portador de código iVProg para linguagem como C ou Java.

O iVProg está disponível no endereço <http://www.matematica.br/ivprog> e pode utilizado no próprio navegador da internet ou descarregado para ser executado como aplicativo.

5.2 Contribuições

O iVProg, um dos resultados desta dissertação, apresenta uma série de adaptações e melhorias feitas no sistema *Alice* que é desenvolvido em Java, na forma de *software livre*, entretanto apresentando sérias limitações para seu uso via Web em escolas brasileiras: ele não está codificado de modo internacionalizável e seu tamanho em bytes é atualmente inviável para uso via internet (*applet* que funciona em navegadores).

Deste modo, o maior trabalho até aqui no iVProg foi adaptar o modelo do *Alice* para um sistema que funcione na forma de *applet* e que esteja em Português (internacionalizável). Para isso foi necessário eliminar um dos recursos do *Alice* que foi sua interface que permitia construção de histórias animadas.

As duas contribuições mais importantes do iVProg estão relacionadas à redução de dificuldades encontradas por alunos em cursos tradicionais de introdução à programação. A mais importante é que o iVProg reduz significativamente as dificuldades de alunos com a sintaxe rígida (e não natural) de linguagens tipicamente utilizadas em cursos introdutórios de programação (como *Java* ou *C*).

A segunda contribuição deste projeto é reduzir substancialmente a curva de aprendizado de um ambiente de programação, pois como foi mostrado no capítulo 4, sua interface permite um rápido aprendizado da sintaxe necessária à programação (usando o modelo de “arrastar blocos”) e praticamente elimina as dificuldades de instalação do ambiente de programação.

Disponibilizar o iVProg como um *widget* foi motivado pela ideia de trazer a programação para a Web, possibilitando o compartilhamento de programas/algoritmos. A utilização e desenvolvimento de *widgets* no contexto do ensino de programação é raramente feito, por outro lado indivíduos e instituições de ensino demandam ferramentas colaborativas que possam ser facilmente difundidas em redes sociais.

5.3 Trabalhos Futuros

Existem várias possíveis melhorias pretendidas para o iVProg, a principal delas seria um avaliador automático, que poderá ajudar tanto a professores, quanto a alunos. O modelo pretendido baseia-se na construção de um *gabarito*: um código-exemplo e algumas entradas com suas respectivas saídas. Futuramente também esperamos implementar um exportador de código iVProg para linguagem como C ou Java, conforme sugerido por alunos no capítulo 4.

Do ponto de vista didático, pretendemos brevemente testar o iVProg em turma do ensino médio. Estamos interessados em saber se o modelo visual permite uma melhor compreensão deste público sobre o conceito de algoritmo matemático. Esperamos que isso melhore inclusive a compreensão geral deste aluno sobre a Matemática. Destacamos que o iVProg pode ser utilizado no projeto iniciado por PEREIRA (2005), aprimorando a metodologia de ensino de programação em escolas do ensino médio.

Além destes, enxergamos como trabalhos futuros a melhoria de sua estrutura interna e incorporação de novas funcionalidades, como possibilidade de trabalho colaborativo. Para isso será necessário a criação de um modelo de interação que permita o trabalho sincronizado de várias pessoas.

Apêndice A

Manual do iVProg

Durante o curso de Introdução à Programação é necessário que o tempo gasto para aprender a usar o iVProg seja o menor possível. Por esse motivo a sua interface é gráfica simples, para que o usuário possa rapidamente se acostumar com os componentes do sistema.

A tela principal do sistema é composta pelos seguintes módulos que serão detalhados a seguir: Menu do sistema; Botões de execução e edição, Aba de variáveis, Aba de métodos, Implementação do método, Aba de funções. A interface gráfica do sistema no modo aplicativo pode ser vista na figura A.1.

A.1 Menu do sistema

O menu do sistema é composto por 3 itens no modo aplicativo: Arquivo, Exercício e Ajuda.

No modo *on-line* o sistema não disponibiliza o item *Arquivo* pois o applet não possui acesso ao sistema de arquivo do computador do usuário.

Arquivo

Este menu pode ser acessado com a tecla de atalho Alt+F. Ao acessá-lo, são disponibilizados os submenus conforme ilustra a figura A.2

O submenu **Novo** cria um novo arquivo com extensão *a2w* reconhecido pelo iVProg.
Tecla de atalho: *Ctrl+N*.

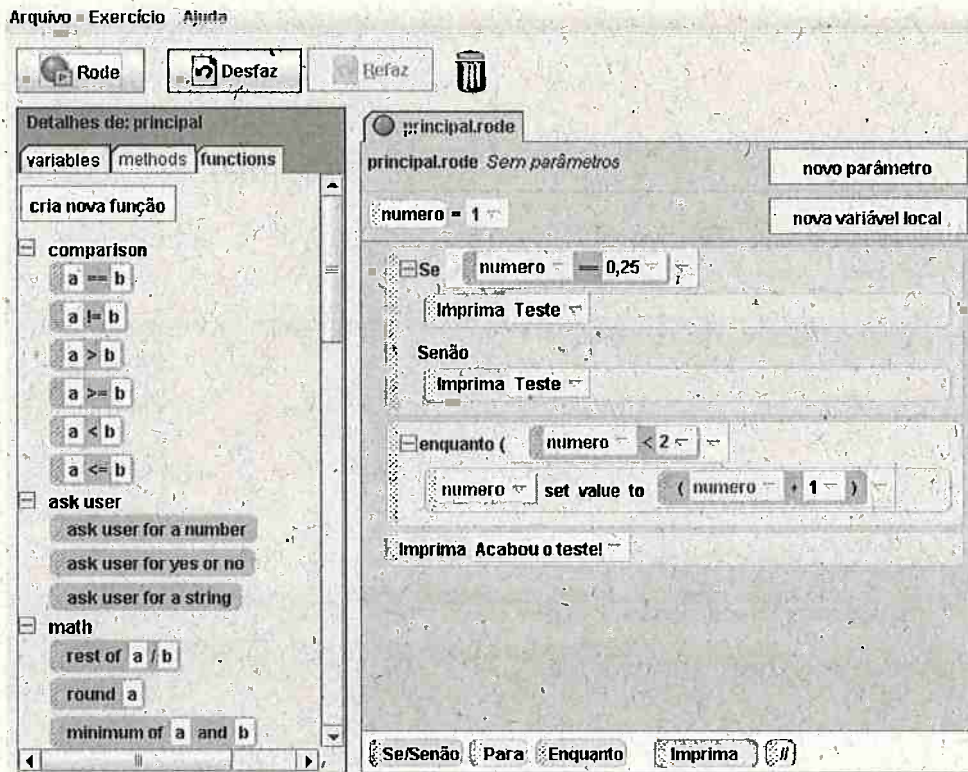


Figura A.1: Tela do iVProg.

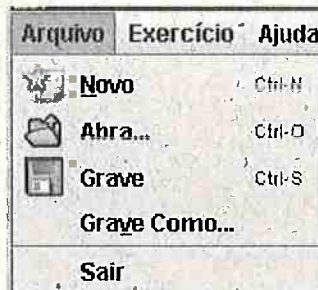


Figura A.2: Menu arquivo

O submenu **Abra** abre arquivo com extensão *a2w*. Tecla de atalho: *Ctrl+O*.

O submenu **Grave** grava um arquivo com extensão *a2w*. Tecla de atalho: *Ctrl+S*.

No submenu **Grave como...** é possível gravar arquivos com outros nomes e outras extensões como por exemplo *xml*. Não há tecla de atalho.

O submenu **Sair** sai e fecha o iVProg. Caso o arquivo não foi salvo, o sistema pergunta ao usuário se deseja salvá-lo. Não há tecla de atalho.

Exercício

Utilizado por alunos ou professores que queiram disponibilizar o seu algoritmo em um Sistema Gerenciador de Cursos (SGC) através do formato *xml* ou disponibilizá-lo na internet em uma página no formato *html*. Este menu pode ser acessado com a tecla de atalho Alt+E. Ao acessá-lo, são disponibilizados os submenus conforme ilustra a figura A.3

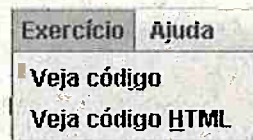


Figura A.3: Menu exercicio

O submenu **Veja código** pode ser acessado para verificar o exercicio no formato *xml* e enviar a um SGC. Não há tecla de atalho para esta funcionalidade.

O submenu **Veja código HTML** pode ser acessado para gerar uma página no formato *html* e disponibilizar o programa na internet. Não há tecla de atalho para esta funcionalidade.

Ajuda

Este menu pode ser acessado com a tecla de atalho Alt+H. Ao acessá-lo, é disponibilizado apenas um submenu conforme ilustra a figura A.4

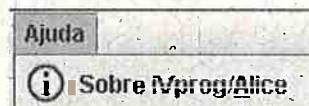


Figura A.4: Menu ajuda

O submenu apenas informa a versão do iVProg e fornece a URL para obter o manual *on-line* e todas informações sobre o sistema.

A.2 Botões de execução e edição

Na parte superior da tela principal, são disponibilizados estes 3 botões e um ícone de 'Lixo' como ilustra a figura A.5.

O botão **'Rode'** é utilizado para executar o programa/algoritmo do usuário. Conforme descrito nos capítulos anteriores não é necessário compilar e sim apenas executar o programa através deste botão.



Figura A.5: Botoes e o icone 'Lixo'.

O botão 'Desfaz' é utilizado para desfazer a última ação executada pelo usuário. Pode ser acessado com a tecla de atalho *Alt+U*.

O botão 'Refaz' é utilizado para refazer a última ação executada pelo usuário. Pode ser acessado com a tecla de atalho *Alt+R*.

O ícone 'Lixo' é utilizado para remover funções, variáveis ou blocos do programa. Esta funcionalidade pode ser acessada pela tecla *Del*.

A.3 Aba de variáveis

Esta aba contém a declaração das variáveis globais e o botão para a criação de uma nova variável.

Variáveis globais são variáveis que podem ser acessadas e modificadas por qualquer método ou função do programa.

O botão 'Nova variável global' é utilizado para criar novas variáveis. Estas variáveis podem ser do tipo numérico, booleano, texto (string) ou lista de qualquer tipo citado anteriormente.

A.4 Aba de métodos

Esta aba contém todos os métodos definidos pelo usuário. Por padrão, todo programa contém o método *rode* criado automaticamente pelo iVProg e é chamado quando o usuário executa o programa.

O usuário pode implementar outros métodos através do botão *cria novo método* como ilustra a figura A.6.

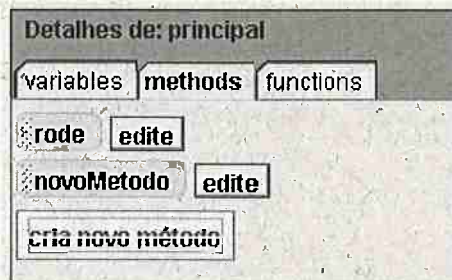


Figura A.6: Aba de métodos.

A.5 Implementação do método

Um método pode possuir variáveis locais, ou seja, variáveis que podem ser acessadas ou modificadas apenas no método no qual foram definidas. O seu escopo é restrito ao método.

Estas variáveis podem ser criadas através do botão *nova variável local* conforme ilustrado na figura A.7

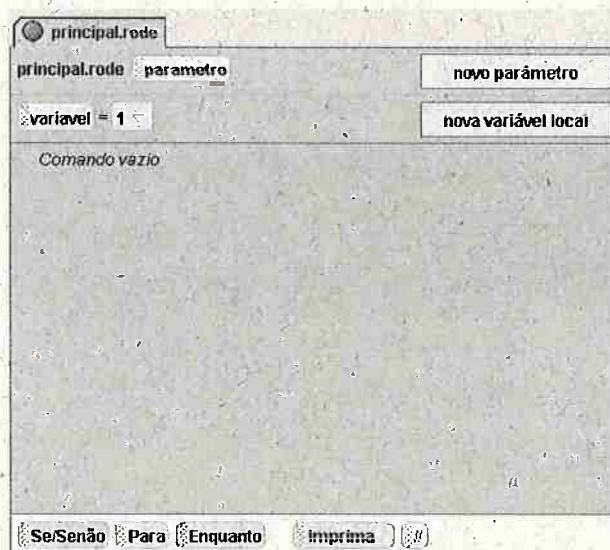


Figura A.7: Aba do método no iVProg

Logo abaixo da declaração das variáveis locais, está a implementação do método. Esta é a região que pode conter variáveis, parâmetros, métodos, funções e inclusive chamada recursiva, ou seja, a função pode fazer uma chamada para a própria função. Segue um exemplo de um método que imprime a somatória dos números naturais menores que n , número digitado pelo usuário.

Esta variáveis podem ser criadas através do botão *nova variável local* conforme ilustrado na figura A.8

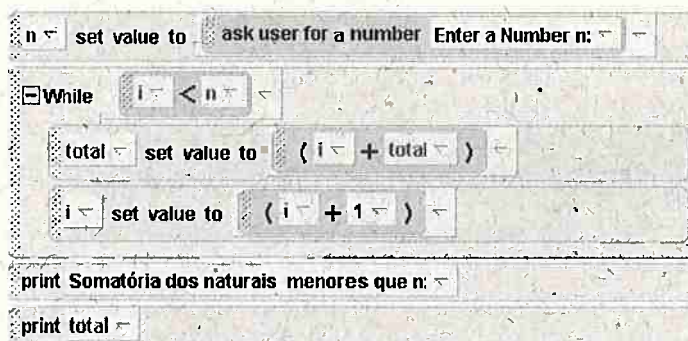


Figura A.8: Exemplo de implementação de método.

Uma observação importante é que ao selecionar algum componente com o mouse, as regiões destacadas em amarelo demonstram os locais em que o componente pode ser utilizado. E a borda verde indica o local em que o componente será utilizado se o usuário soltar o botão do mouse. Esta situação é exemplificada na figura A.9

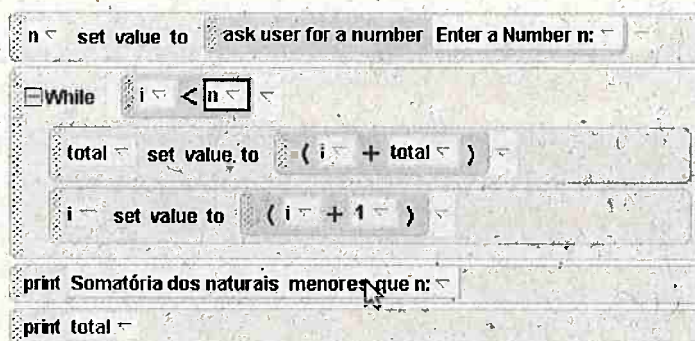


Figura A.9: Destaque das cores em locais em que o componente selecionado pode ser utilizado.

A.6 . Aba de funções

Esta aba contém todas as funções definidas pelo usuário. No iVProg, funções são componentes similares aos métodos, entretanto, retornam um valor do tipo numérico, booleano ou *string* conforme definido pelo usuário.

Por padrão, o sistema disponibiliza várias funções de comparação, operações algébricas, trigonometria, lógica booleana, manipulação de textos e outros conforme ilustra a figura A.10.

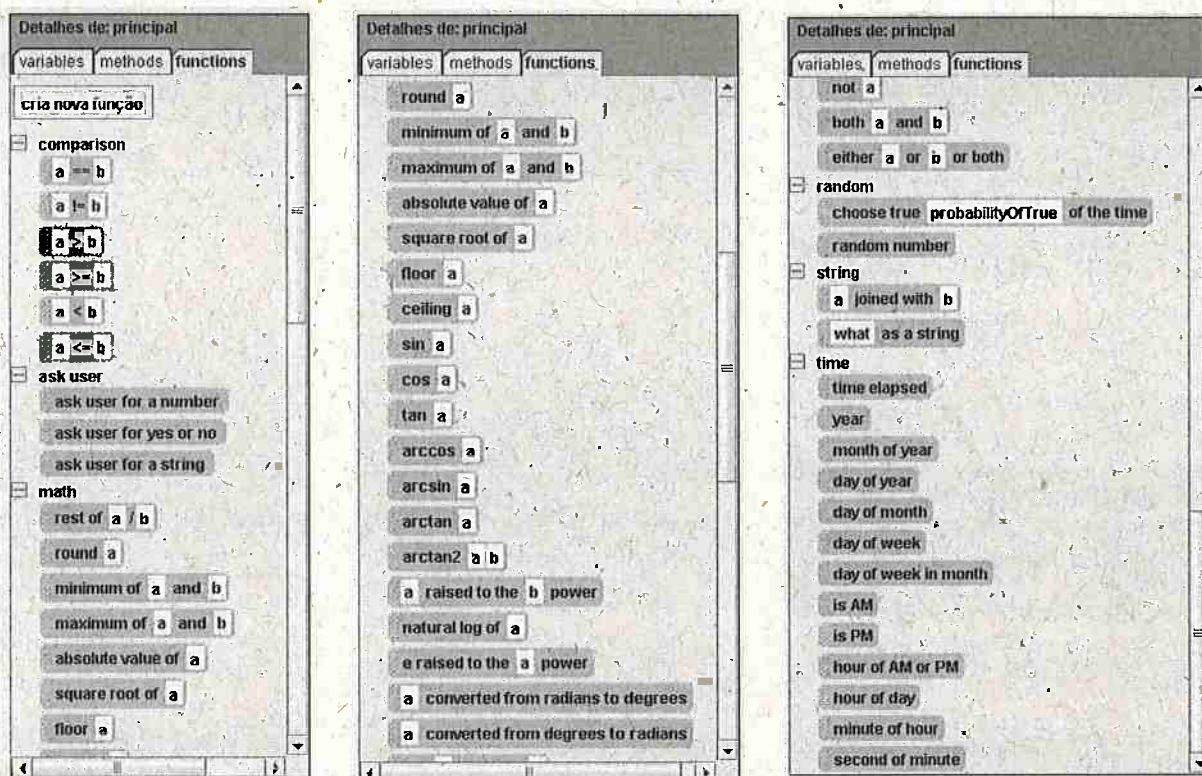


Figura A.10: Funções do iVProg.

Apêndice B

Relatos obtidos nos questionários

A seguir são apresentados todos os comentários feitos pelos alunos nos questionários após o uso do iVProg em disciplina regular descrito na seção 4.2. Estes comentários estão documentados, por escrito, nos próprios questionários. Com o objetivo de manter a fidelidade na transcrição dos comentários, nenhuma correção ortográfica ou gramatical foi feita sobre o texto original.

1) Por favor, destaque as principais vantagens de programar com o iVProg em relação à linguagem C:

“Todos os comandos que em C devem ser digitados, no iVprog basta que arrastemos o ícone. O que torna mais hábil para manuseio.”

“Maior interatividade e a não necessidade de decorar códigos. A parte visual ajuda muito a construir o raciocínio.”

“Eu acho que a principal vantagem de usar o iVprog foi não precisar ficar preocupado com a sintaxe dos comandos. Isso com certeza nos ajuda a dar atenção ao algoritmo em questão.”

“Deixa claro o início e o fim de um laço pelo fato de utilizar ‘caixas’. Não exige que o usuário decore funções de comando, já que todas as ferramentas e funções estão apresentadas na interface do programa.”

“A vantagem é que você não precisa se preocupar com a linguagem, bastando entender a lógica. Logo, para um primeiro contato com um exercício e mais vantajoso pensá-lo utilizando o iVprog, tem ainda a vantagem de diminuir riscos de erro de digitação.”

“Visualização dos algoritmos, biblioteca de funções de fácil acesso previamente definidas”

“A não necessidade de ”decorar” comandos - A organização em blocos oferecida pelo iVprog - Atratividade visual - Comandos em Português”

“Em C é necessário usar um compilador e digitar comandos no terminal do Linux, o que torna lento o processo de teste do programa. a interface do Emacs eh extremamente confusa pra um programa tao simples.”

“Não tem risco de errar na digitação. Basta só pensar em algoritmo.”

“No iVProg, nos preocupamos menos com a sintaxe em si, dando mais atenção para a logica da resolução.”

“Não da para errar na hora de digitar o código isso é bom. e criar duas funções é mais fácil no ivprog.”

“Estruturar o raciocínio do algoritmo.”

“Com o iVProg tenho opções para ir tentando fazer o exercício enquanto em C, senão tiver noção de como começar, fica difícil partir do zero.”

“O iVprog é melhor para tratar a parte matemática, ao invés de se preocupar com a linguagem”

“Sintaxe pronta, controle de variável, alteração no nome da variável altera em todo o programa.”

“Acho que a maioria das pessoas tem dificuldade na sintaxe em C, como no iVProg, não se digita nada, se torna mais fácil.”

“Os blocos de comandos ficam bem visíveis e fácil de entender.”

“Eu comecei um curso de ”Introdução a computação”, que foi dado inteiramente em c++. Particularmente achei muito ruim pois já começava de modo complexo e formalizado, e visto que era um curso de engenharia, não precisaria ter tanta ênfase nas formalidades de programação. Ao começar o curso de MAC110, achei muito bacana a forma de trabalhar com o iVprog, visto que esta é a INTRODUCAO de computação nesse curso, eh o primeiro contato que temos com a programação. O iVprog trabalha de uma forma extremamente didática. Ele não é SUPER simples, mas certamente muito melhor do que iniciar o curso em c. Para o curso de licenciatura em particular, o trabalho com o iVprog eh muito mais eficiente, pois o curso visa aplicar conhecimentos matemáticos a computação, e não nos tornar profissionais no assunto.”

2) Por favor, destaque as principais vantagens de programar usando a linguagem C em relação ao iVProg:

“Velocidade, formalização de dados e maior liberdade para o usuário organização desses dados.”

“As vantagens são que você estará usando uma linguagem em que é possível fazer programas, e não só entendendo algoritmos. Assim você começa a entender mais o computador, seu funcionamento, as bibliotecas padrão etc. ”

“Rapidez em rodar um programa maior; *poder copiar e colar algum programa pronto, útil para programas muito longos; *melhor para compilar.”

“Não trava, liberdade, mais fácil construção de algoritmos”

“Corremos menor risco de perder o que foi programado. A compilação do c é mais clara com relação aos erros de sintaxe e até mesmo de lógica. ”

“Possibilidade de copiar blocos de comandos para possível reaproveitamento”

“Identificar a linha na qual o erro se encontra; menos medo de perder tudo o que você está fazendo no processo de envio.”

“A linguagem C cobrou mais organização, para que pudéssemos compreender a sintaxe.”

“A linguagem C possibilita um leque muito maior de resoluções e dá muito mais liberdade de criação de algoritmos e tem várias funções que o iVprog não tem. Além do mais, o processamento na linguagem C é correto e muito mais rápido.”

“Se habituar com a linguagem de programação.”

“Não enxergo vantagens porque creio que o iVProg tem uma compreensão um pouco mais fácil.”

“A linguagem C permite que seja visto o ”corpo” do programa, o que já não é possível no iVprog”

“Desconheço, talvez a velocidade para compilar alguns programas mais sofisticados.”

“Em c, podemos encontrar o erro mais facilmente e aprendemos de modo mais abstrato a programar”

“Dispensa o uso do mouse, você consegue visualizar o programa inteiro. A cada erro de compilação cometido você consegue identificar o tipo e o local do erro, e acostuma a

não cometê-lo mais vezes. A linguagem digitada e o tipo de linguagem usada em todos os lugares.”

“Flexibilidade, independência do navegador.”

“Em minha opinião, a única vantagem que da linguagem C e a velocidade para fazer cálculos. Mas em suma, e muito complicado de se mexer por conta da sintaxe.”

“Não ha erros e o programa é executado mais rápido.”

“As únicas vantagens de programar em linguagem C e que como esta linguagem e mais conhecida universalmente, encontra-se facilidade de achar livros e informações em sites a respeito dessa linguagem e os programas de linguagem C em geral não travam!”

“Em C, podemos utilizar compiladores ou editores de maneira mais flexível. Já o iVprog na versão web restringe um pouco seu uso, uma vez que seu bom funcionamento depende de fatores tais quais a qualidade da conexão, do próprio computador, etc. A versão aplicativa e mais flexível.”

“No iVprog muitas vezes perdíamos o que já tinha sido feito em função de algum problema no programa em si. Já em C isso não ocorre.”

“Certamente para um trabalho profissional a linguagem C é muito mais eficiente. Ela possibilita você utilizar uma quantidade enorme de dados em um curto período de tempo. Ela da também um caráter de formalização, organização dos conteúdos, que é sim muito importante para o curso, mas não nas primeiras semanas.”

3) Existe algum recurso que gostaria de ver implementado no iVProg? (em caso afirmativo favor citar qual)

“Acho que o iVProg já é um programa bom e não necessita de implementações, porem acho que o curso deveria escolher ou usar o iVProg, ou usar linguagem C, pois a transição de um para outro se torna muito mais difícil.”

“Poder abrir o programa feito pela iVprog diretamente pelo programa, e não por meio do iVprog. Como em c com .exe.”

“Quanto estávamos no aplicativo, o programa nos alertava de tempo em tempo para salvar o que foi feito, já online isso não acontece e se o programa travar ou acontecer qualquer outro imprevisto perdemos tudo. Talvez se houvesse um jeito de salvar online seria bom!”

“Os textos no iVprog [prints] não estão muito legais, pois você não pode fazer adequações para o programa ficar bem visto, exemplo, : ”O Numero 'x' esta na coluna 'y' ”, etc”

“Independência dos laços de if/else, já que, nem sempre é necessário utilizarmos o else na função if.”

“Sim. Seria interessante que os programas feitos com o iVProg gerassem um arquivo executável.”

“Não, pois acho que o iVprog esta muito bom agora. Tem de se preocupar em resolver os bugs que já são conhecidos”

“Gostaria que o iVProg fosse em inglês, com os mesmos termos da linguagem em C.”

“Vetores, matrizes e poder mudar a strings para "Enter a number"”

“Sim. um recurso que apontasse ou desse uma dica do erro cometido”

“Criação de uma área comum para armazenamento de funções para reaproveitamento em outro código. O padrão de sempre iniciar sem nada não facilita a reutilização objetos.”

“Talvez o principal problema encontrado por todos no iVprog, foi o travamento exagerado quando tentávamos rodar o programa. E como algumas vezes esse erro se dava a alguma janela aberta, acredito que poderia ser impresso na tela um aviso indicando que a janela ainda estava aberta. Também acredito que se os exercícios pudessem ser salvos quando o fazemos de modo online facilitaria muito, pois como eles não podiam ser entregues em arquivos compactados, tínhamos que fazê-los na internet, e muitas vezes perdíamos o que havíamos feito por causa de algum problema que o iVprog apresentava.”

“Poder salvar online o programa que esta sendo criado, para não correr o risco de perder tudo. Um programa destinado ao aprendizado não deveria travar quando um aluno faz um erro de programação. Naturalmente num curso de introdução é normal que os alunos cometam erros e se o programa simplesmente trava, não sabemos onde estamos errando nem que tipo de erro estamos cometendo. Acho que o iVProg poderia ter um tipo de "manual" com linguagem para leigos entenderem.”

“Os valores iniciais poderiam incluir 10 e não incluir 0.25 nem 0.5. modo "complicado" da função for esta descrito de modo confuso; rodar o programa ainda necessita de varias tentativas; copiar e colar é possível mas não está indicado; poderia poder colocar funções após o "return" ao criar uma função (porque faz parecer que o programa travou) Um bug constante era que o iVprog aparentava funcionar mas ao tentar trocar o parâmetro de uma função ou usar "math" em qualquer variável, o conteúdo daquele campo permanecia inalterado (isso só era solucionado apos fechar o programa) função break (não me lembro se tinha) tentei por muito tempo fazer vetores mas não consegui, apesar de que aparentemente existia uma

opção. operações matemáticas poderiam ser implementadas de outra forma (talvez um menu separado com operações, assim como há um menu só para funções e outro só para variáveis globais)”

Referências Bibliográficas

- ALICE (2010). *Alice - Learn Programming*. <http://www.alice.org/>: Carnegie Mellon University.
- ALMEIDA, M. E. B. (2001). Tecnologia na escola: Formação de educadores na ação. *Programa Salto para o Futuro-série Informática na Educação*, 1(1), 2-7.
- AMBAP (2010). *AMBAP - Ambiente de Aprendizado de Programação*. <http://www.ufal.br/tci/ambap>: Universidade Federal de Alagoas (UFAL), Centro de Ciências Exatas e Naturais (CCEN).
- BAEZA, R. A. Y. (1995). Teaching algorithms. *International organization that fosters and promotes the discovery and dissemination of high quality research in theoretical computer science - SIGACT* 26(4), 51-59.
- BLUEJ (2010). *BlueJ - The interactive Java environment*. <http://www.bluej.org>: University of Kent.
- BORGES, M. A. F. (2000). Avaliação de uma metodologia alternativa para a aprendizagem de programação. *VIII Workshop de Educação em Computação - WEI*, 1-7.
- BRANDÃO, L., S. ISOTANI, & J. G. MOURA (2006). Imergindo a geometria dinâmica em sistemas de educação a distância: igeom e saw. *Revista Brasileira de Informática na Educação* 1(14), 41-49.
- BURTON, P. J. (2003). Teaching Programming in the OOP Era. *Special Interest Group on Computer Science Education (SIGCSE)* 35(2), 11-14.
- CAMPOS, R. L. B. L. (2009). Lógica de programação: Há como melhorar o aprendizado fugindo dos padrões estabelecidos nos livros didáticos e adotados pela maioria dos docentes? *XVII Congresso Iberoamericano de Educacion Superior em Computacion*, 2-5.

- CHAVES DE CASTRO, T. (2003). Utilizando programação funcional em disciplinas introdutórias de computação. *XI Workshop de Educação em Computação WEI 2003*, 1-7.
- CIDRAL, A. (2000). Aps-tutor: Um ambiente web de aprendizagem como suporte às atividades presenciais no ensino de análise de sistemas. *V Congresso Iberoamericano de Informatica Educativa - RIBIE*, 1-7.
- DELGADO, C. (2004). Uma abordagem pedagógica para a iniciação ao estudo de algoritmos. *XII Workshop de Educação em Computação*, 2-7.
- FALCKEMBACH, G. A. M. & F. V. DE ARAUJO (2006). Aprendizagem de algoritmos: dificuldades na resolução de problemas. *Universidade Luterana do Brasil (ULBRA), Santa Maria, RS.*, 1-5.
- GAMEMAKER (2010). *Game Maker*. <http://www.gamemaker.nl>: YoYo Games Inc.
- GOERTZEL, K. (2009). Introduction to software security. *Department of Homeland Security and Department of Defense Data and Analysis Center for Software*, 2-4.
- GREENFOOT (2010). *Greenfoot*. <http://www.greenfoot.org>: Universidade de Kent.
- HARA, N. & R. KLING (1999). Student's frustrations with a web-based distance education course. *First Monday: Journal on the Internet*, 4-12.
- HENDERSON, P. B. (1986). Anatomy of an introductory computer science course. *In Proceedings of the seventeenth SIGCSE technical symposium on Computer science education Special Interest Group on Computer Science Education - SIGCSE(1)*.
- HENDERSON, P. B. (1987). Modern introductory computer science. *In Proceedings of the eighteenth SIGCSE technical symposium on Computer science education (1)*, 183-190.
- HENTEA, M., M. SHEA, & L. PENNINGTON (2003). A perspective on fulfilling the expectations of distance education. *Conference On Information Technology Education*, 160-167.
- JELIOT (2010). *Jeliot*. <http://cs.joensuu.fi/jeliot/>: Joensuu Yliopisto - Weizmann Institute of Science.
- Kahn, K. (2010). *ToonTalk - Making programming child play*. <http://www.toontalk.com>.
- KOLIVER, C. (2004). Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *XII Workshop de Educação em Computação*, 1-5.

- LEMOS, M. A. (2003). Uma biblioteca cognitiva para o aprendizado de programação. *XI Workshop de Educação em Computação - WEI*, 1-3.
- LITTO, F. M. & A. FILATRO (2004). Brazilian research on distance learning. *A state-of-the-art study. In Proceedings of International Congress of Distance Education*, 2-6.
- MANSO, E. (2010). *PORTUGOL*. <http://orion.ipt.pt/~manso/Portugol/>: Instituto Politécnico de Tomar - Escola Superior de Tecnologia.
- MARCZAK, S. d. S. & L. M. M. GIRAFFA (2003). Ambientes inteligentes para suporte ao ensino de programação. *Technical Reports Series* (28), 2-3.
- MARION, W. (1999). Cs1: What should we be teaching? *Annual Joint Conference Integrating Technology into Computer Science Education, Working group reports from ITiCSE on Innovation and technology in computer science education*, 35-38.
- MATSUDA, N. & K. VANLEHN (2004). Grammy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning* 32(1), 3-33.
- MOODLE (2010). *Moodle: open-source community-based tools for learning*. <http://moodle.org/>.
- MOURA, J. G. (2007). Saw - sistema de aprendizagem pela web: motivações e desenvolvimento. Dissertação (mestrado em ciências da computação), Universidade de São Paulo - USP.
- NIELSEN, J. (2000). Why you only need to test with 5 users. *Alertbox*.
- NOBRE, I. A. (2002). Suporte à cooperação em um ambiente de aprendizagem para programação (samba). *XIII Simpósio Brasileiro de Informática na Educação - SBIE*, 1-9.
- OLIVEIRA, A. L. (2005). Avaliação comparativa de diferentes modelos de interfaces gráficas empregadas no ensino de geometria, segundo os conceitos de usabilidade. Dissertação de mestrado, Universidade de São Paulo.
- PEREGO, C. (2002). A migração de pascal para java: Problemas e propostas de solução, x workshop de educação em computação. *X Workshop de Educação em Computação - WEI*, 1-9.
- PEREIRA, J. (2005). Ensino de algoritmos e programação: uma experiência no nível médio. *XXV Congresso da Sociedade Brasileira de Computação, Unisinos, São Leopoldo, RS, Brasil*, 1-8.

- RAABE, A. L. A. R., R. DE SANTIAGO, & R. L. S. DAZZI (2007). Adquirindo experiência na construção de ferramentas de apoio a aprendizagem de algoritmos. *Workshop de Ambientes de Apoio a Aprendizagem de Algoritmos e Programação. Simpósio Brasileiro de Informática na Educação (XVIII)*, 1-5.
- RAPKIEWICZ, C. (2006). Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *Novas Tecnologias na Educação 4(2)*, 2-9.
- RAYMOND, E. S. (2000). *The Cathedral and the Bazaar* (2 ed.). London: OReilly Media.
- RODRIGUES, M. C. (2005). Como ensinar programação? *Informática - Boletim Informativo 1(1)*, 1-5.
- RODRIGUES, P. A. & L. O. BRANDÃO (2009). Tarefa interativa: uma proposta flexível de interatividade para o moodle. *Simpósio Brasileiro de Informática na Educação (SBIE 2009)*, 1-8.
- SANTOS, G. (2003). Autoria e interpretação tutorial de soluções alternativas para promover o ensino de programação de computadores. *XIV Simpósio Brasileiro de Informática na Educação - SBIE 2003*, 2-7.
- SANTOS, R. P. & H. A. X. COSTA (2006). Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *Journal of Computer Science 5(1)*, 1-6.
- SCRATCH (2010). *Scratch is a new programming language*. <http://scratch.mit.edu/>: Massachusetts Institute of Technology (MIT).
- SUN (2010). *Java Look and Feel Guideline*. <http://java.sun.com/products/jlff/ed1/dg/higf.htm>: SUN.
- TOBAR, C. M. (2001). Uma arquitetura de ambiente colaborativo para o aprendizado de programação. *XII Simpósio Brasileiro de Informática na Educação (SBIE-2001)*. Vitória, ES, Brasil, 1-6.
- VALENTE, J. A. Informática a educação no brasil: Análise e contextualização histórica. *O computador na sociedade do conhecimento UNICAMP/NIED*, 1-8.
- VISUALG (2010). *VisuAlg*. <http://www.apoioinformatica.inf.br/visualg/>: Apoio Informática Ltda.

- WEBPORTUGOL (2010). *Webportugol: ferramenta para ajudar aos alunos a fazerem seus primeiros algoritmos*. <http://www.univali.br/webportugol>: Universidade do Vale do Itajaí - UNIVALI.
- WESLEY, H., A. P. GONDIM, & A. P. AMBROSIO (2008). Esboços de fluxogramas no ensino de algoritmos. *Workshop sobre Educação em Computação WEI. Anais do XXVIII Congresso da Sociedade Brasileira de Computação*, 1-4.
- WINSLOW, L. E. (1996). Programming pedagogy - a psychological overview. *Special Interest Group on Computer Science Education - SIGCSE 28(3)*.