Desenvolvimento de um Interpretador LOGO Multi-Plataforma

Eduardo Teodoro Silva Júnior

Lavras - 2002

Eduardo Teodoro Silva Júnior

Desenvolvimento de um Interpretador LOGO Multi-Plataforma

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. MSc. Bruno de Oliveira Schneider

Co-Orientador
Prof. MSc. Joaquim Quinteiro Uchôa

Lavras
Minas Gerais - Brasil
2002

Eduardo Teodoro Silva Júnior

Desenvolvimento de um Interpretador LOGO Multi-Plataforma

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em *01 de Abril de 2002*

Prof. DSc. José Monserrat Neto

Prof. MSc. Bruno de Oliveira Schneider
(Orientador)

Prof. MSc. Joaquim Quinteiro Uchôa
(Co-Orientador)

Lavras
Minas Gerais - Brasil

À toda minha família,
à minha namorada Cecília,
e especialmente aos meus pais
Eduardo ($In\ Memorian$*) e Ângela.*

Agradecimentos

Agradeço aos meus pais que são minha motivação e inspiração por toda a vida.

Agradeço aos meus irmãos Júlio César, Paulo Ernane, Vagner e João Donizete pelo incentivo que sempre me deram e pela confiança que tiveram no meu potencial.

Agradeço aos meus colegas de apartamento Alexandre, André, Sérgio e Wanner pela ótima convivência, apoio e amizade.

Agradeço aos meus amigos Mário Luiz, Marcos Aurélio, Talles, Mário Filho e Dênis pelo apoio, confiança e amizade.

Agradeço ao Prof. Bruno Schneider e Prof. Joaquim Uchôa pelo apoio e incentivo durante a orientação deste trabalho.

Resumo

A linguagem LOGO vem sendo adotada com sucesso em várias escolas e outras instituições de ensino desde a segunda série do primeiro grau até cursos de graduação e pós-graduação. Apesar do LOGO ser uma linguagem bastante popular, não existe um interpretador para o sistema operacional *Linux* que seja fácil de encontrar. Nos últimos anos o sistema *Linux* vem se difundindo por todo o mundo e se tornando uma alternativa mais segura, confiável e viável economicamente que o sistema *Windows* da *Microsoft*. Percebe-se facilmente que as escolas públicas brasileiras sofrem com a falta de recursos financeiros, e poderiam economizar um grande montante adotando o sistema *Linux*, além de evitar problemas com o uso de *software* não licenciado, infecções por vírus e danos à configuração do sistema por usuários inexperientes. O objetivo deste projeto é desenvolver um interpretador LOGO gratuito e que esteja disponível para os dois sistemas operacionais supracitados, além de facilmente adaptado para sistemas como MacOS, Solaris, AIX, HP e diversas variantes do Unix. Isto possibilitará que usuários de interpretadores LOGO tenham liberdade de escolher seu sistema operacional.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O uso do computador na educação é uma realidade cada vez mais comum e parece ser inevitável, o computador invadiu praticamente todos os campos profissionais e ocupacionais, e com o processo educacional não pode ser diferente. O computador é uma poderosa ferramenta que pode mudar radicalmente o sistema de ensino como conhecemos, e isso implica em medos e críticas severas. Apesar das visões futuristas e alarmistas, se torna cada vez mais claro, que o papel do computador é tornar-se um meio auxiliar de ensino, um recurso a mais para a diminuição das carências desta área.

O computador em si não pode ajudar em nenhuma tarefa sem um *software* adequado, isso acontece também na educação. O que se pode verificar atualmente é que o computador tem sido sub-utilizado pela maioria das instituições de ensino, que muitas vezes apenas ensinam seus alunos a operar os sistemas e aplicativos mais comuns, ou automatizam os métodos tradicionalmente adotados, sem fazer uso de toda a capacidade potencial desta máquina, para isso faz-se necessário o uso de um *software educacional*.

Um dos mais antigos e populares *software* educacional do mundo, é o interpretador para linguagem LOGO. LOGO é uma linguagem de programação desenvolvida no MIT (Massachussets Institute of Technology) na década de 60, por uma equipe comandada pelo matemático Seymour Papert, o qual é adepto convicto das idéias da pedagogia construtivista de Jean Piaget. A linguagem LOGO é simples e seus comandos básicos podem ser aprendidos e usados por uma criança de 8 anos, além disso o LOGO é atrativo pois seu ambiente assemelha-se a um jogo de computador. O objetivo da programação é movimentar uma tartaruga virtual, e ensiná-la a fazer desenhos cada vez mais complexos dependendo da idade e ca-

pacidade do aprendiz. Segundo Papert quando uma criança aprende a programar o processo de aprendizagem é transformado, ela se torna mais ativa e capaz de guiar-se sozinha.

A UFLA oferece o curso de pós-graduação *lato sensu*: *Informática em Educação*, no qual é ensinado a linguagem LOGO e sua filosofia. No decorrer do curso os professores notaram a dificuldade em se encontrar um interpretador LOGO para o sistema *Linux*, o que obriga a instituição que deseje adotar o *software* a usar o sistema *Windows*, o qual possui um alto custo, principalmente para escolas públicas que sofrem com a falta de recursos financeiros, mesmo para adquirir os itens básicos necessários à ministração de aula, e por desconhecerem a existência de sistemas gratuitos muitas vezes usam sistemas sem a licença adequada.

O objetivo deste projeto é desenvolver um interpretador para a linguagem LOGO, em português, que seja multi-plataforma, ou seja, que possa ser usado em mais de um sistema operacional, neste caso testado para *Linux* e *Windows* com possibilidade de trabalhar em MacOS, Solaris, AIX, HP e outras variantes do Unix. Será mantida a compatibilidade em relação à linguagem básica, e aos comandos mais importantes, implementados pelos interpretadores LOGO mais tradicionais disponíveis gratuitamente em português. O interpretador desenvolvido será distribuído gratuitamente sob a licença GPL, que protege *software* de código aberto.

Esta monografia está organizada da seguinte forma: No capítulo 2 são apresentados os subsídios teóricos necessários para a boa compreensão da filosofia LOGO, são discutidos o construtivismo, construcionismo e seus idealizadores. No capítulo 3 é abordada a importância do computador para a educação e como ele deve ser utilizado para se atingir bons resultados, além da apresentação de possibilidades para tornar mais viável sua adoção, como o uso do sistema *Linux* e *software* sob licença GPL. No capítulo 4 é apresentada a linguagem de programação LOGO, sua origem, influências, características, vantagens e os comandos mais usados. No capítulo 5 é proposto e apresentado o wxLogo, o interpretador multi-plataforma para a linguagem LOGO, que foi desenvolvido durante este trabalho, são citadas as melhorias adicionadas em relação a outros interpretadores, os comandos e funções já implementadas e uma breve descrição da interface e suas funcionalidades. No capítulo 6 serão apresentados os recursos utilizados para o desenvolvimento do wxLogo, detalhes sobre a análise e interpretação do código, bem como detalhes sobre a implementação, os principais problemas que precisaram ser contornados e os que ainda não foram solucionados. No capítulo 7 são apresentadas as conclusões sobre o trabalho e sugestões para futuras melhorias.

Capítulo 2

Revisão de Literatura

2.1 Introdução

Para entender a maneira como a linguagem LOGO pode ajudar no desenvolvimento da criança e porque é tão conhecida e elogiada por especialistas em educação, é preciso antes conhecer as idéias que inspiraram o seu desenvolvimento. O objetivo deste capítulo é fornecer uma base teórica para que se possa entender melhor o trabalho que se segue. Serão apresentados os fundamentos e os idealizadores das teorias que inspiraram e influenciaram a criação da linguagem LOGO.

2.2 Construtivismo

A escola tradicional aplica um método chamado *instrucionista*, no qual a criança recebe informações passivamente através do professor e, é avaliada por questionários, onde suas respostas são consideradas como certas ou erradas, sem maiores preocupações com o processo envolvido no desenvolvimento da resposta. Existe uma outra linha de pensamento na pedagogia que defende uma idéia diferente, segundo a qual o aprendiz é responsável por construir seu conhecimento, e é levado a isso através do raciocínio e interação com o meio em que vive, esta corrente de pensamento é chamada *construtivismo*, segundo [Campos (1997)]:

> "Construtivismo é uma corrente teórica empenhada em explicar como a inteligência humana se desenvolve partindo do princípio de que o desenvolvimento da inteligência é determinado pelas ações mútuas entre o indivíduo e o meio. A idéia é que o homem não nasce in-

> teligente, mas também não é passivo sob a influência do meio, isto é, ele responde aos estímulos externos agindo sobre eles para construir e organizar o seu próprio conhecimento, de forma cada vez mais elaborada".

Como pode ser visto em [Correia et alii (1999)] o construtivismo difere da escola tradicional onde o aluno apenas recebe informações e é esperado que todos assimilem o conteúdo da mesma forma e com a mesma facilidade, em suma tentam *padronizar* o conhecimento dos alunos. No construtivismo o aluno passa a ser ativo, ele deve construir seu próprio conhecimento através da transformação do conhecimento já existente.

Estão presentes no contexto do construtivismo:

- A exigência de uma dinâmica interna de momentos discursivos (raciocínio, dedução, demonstração...);
- O entendimento do presente é baseado no passado e dá ao futuro nova construção - nessa aprendizagem o autor reconstrói o conhecimento, e o educador reflete sua prática pedagógica;
- A constante reconstrução do conhecimento.

O pensamento construtivista é fundamentado nas idéias e estudos de dois pensadores, o suíço Jean Piaget e o russo Lev Vygotsky. Apesar de serem da mesma época os dois não chegaram a se conhecer, principalmente devido a problemas políticos pelos quais passavam a URSS, por isso o trabalho de Vygotsky só foi conhecido mais recentemente.

2.2.1 Jean Piaget

Jean Piaget nasceu em Neuchâtel, Suíça, em 1896. Aos 11 anos de idade publicou seu primeiro artigo, aos 22 anos recebeu o título de doutor em biologia, ao longo de sua vida escreveu cerca de 70 livros e 300 artigos sobre psicologia, pedagogia e filosofia. Morreu em 1980 aos 84 anos em Genebra, Suíça. O principal enfoque de seus estudos é entender como o indivíduo se desenvolve psicologicamente, ou seja passa de um estado de menor conhecimento para outro de maior conhecimento. As teorias de Piaget sobre o conhecimento foram denominadas por ele próprio de *epistemologia* por serem centralizadas no conhecimento científico, e de *genética*, porque estudava também as condições necessárias para que o indivíduo chegue na

idade adulta com o desenvolvimento máximo de seu potencial intelectual. Disto, surge o termo *epistemologia genética* ou *psicogenética* [Correia et alii (1999)].

Para Piaget, o indivíduo absorve conhecimento a partir de experiências, ações e interações com o meio, estas experiências vão sendo assimiladas, interiorizadas. Isso constitui um processo permanente de construção e reconstrução das estruturas cognitivas o que resulta na formação da estrutura do pensamento. Esta estrutura se altera cada vez que o indivíduo interage com um novo elemento, fazendo com que este elemento seja assimilado pela estrutura já existente.

Piaget divide o comportamento infantil em quatro estruturas cognitivas primárias, ou estágios de desenvolvimento, mais detalhes em [Correia et alii (1999)] e também em [Campos (1997)], são eles:

- **Sensório-Motor** (0 aos 18/24 meses aproximadamente): nesta fase a criança está explorando o meio físico através de seus esquemas motores.

- **Pré-Operatório** (2 anos a mais ou menos 7 anos): a criança é capaz de simbolizar, evocar objetos ausentes. Estabelece diferença entre significante e significado, o que possibilita distância espaço-temporal entre o sujeito e o objeto, por meio da imagem mental. A criança é capaz de imitar gestos, mesmo com a ausência de modelos.

- **Operatório Concreto** (7 a 11 anos): a criança tem a inteligência operatória concreta, sendo capaz de realizar uma ação interiorizada, executada em pensamento, reversível, pois admite a possibilidade de uma inversão e coordenação com outras ações, também interiorizadas. Necessita de material concreto, para realizar essas operações, mas já está apta a considerar o ponto de vista do outro, sendo que está saindo do egocentrismo.

- **Formal** (entre os 9/10 anos aos 15/16 anos): o adolescente tem as estruturas intelectuais para combinar as proporções, as noções probabilísticas, raciocínio hipotético dedutivo de forma complexa e abstrata.

Baseada na teoria de Piaget o professor deverá ter o cuidado de promover ambientes de aprendizagem ricos e estimulantes com objetos amplos para que as crianças brinquem e aprendam a aprender. Cada estágio tem uma particularidade diferente que deve ser observada para se obter melhores resultados. É importante lembrar que o desenvolvimento deve anteceder o aprendizado, ou seja é importante observar o estágio de desenvolvimento das funções psíquicas e cognitivas de um indivíduo antes de submetê-lo a um determinado processo de aprendizagem.

2.2.2 Lev Vygotsky

Lev Semenovich Vygotsky nasceu em Orsha, Bielorrúsia, em 17 de Novembro de 1896. Formou-se em literatura na Universidade de Moscou, aos 28 anos começou a pesquisar sobre o desenvolvimento psicológico, educação e psicopatologia. Morreu aos 38 anos em 11 de junho de 1934. Sua teoria tem como influência as teorias de Karl Marx, em [Correia et alii (1999)] é citado que a apartir desse embasamento Vygotsky abstrai que o ser humano é criado histórica e socialmente, e que suas relações com a natureza e com os outros homens no nível da consciência são lidados de forma espontânea apenas quando ele não tem percepção da consciência sobre aquilo que está fazendo. Por outro lado, à medida que o homem toma consciência da consciência que possui, mais e mais ele abstrai sobre seus atos e sobre o meio. Com isto, seus atos deixam de ser espontâneos (no sentido biológico do termo) para se tornarem atos sociais e históricos, envolvendo a psique do indivíduo.

Os vygotskianos entendem que os processos psíquicos, a aprendizagem entre eles, ocorrem por assimilações de ações exteriores, interiorizações desenvolvidas através da linguagem interna que permite formar abstrações. Para Vygotsky, a finalidade da aprendizagem é a assimilação consciente do mundo físico mediante a interiorização gradual de atos externos e suas transformações em ações mentais.

Vygotsky se preocupou em descrever e entender o que ocorre ao longo do desenvolvimento algumas funções, assim como, no estudo da linguagem e da formação de conceitos. Nessa teoria não existem estágios de desenvolvimento explicando sobre o surgimento e desenvolvimento das funções psíquicas. Mas existe na teoria de Vygotsky, assim como na de Piaget, os diferentes níveis de funcionamento psicológico, cada qual com características específicas:

- **Pseudo-Conceitos**: Aqui ainda a criança não consegue formular conceitos, mas o pensamento ocorre por cadeia e de natureza fatual e concreta. Nesta fase a criança se orienta pela semelhança concreta visual, formando apenas um complexo associativo restrito a um determinado tipo de conexão perceptual.

- **Conceitos**: Formação de conceitos, atividade complexa e abstrata, que usa o signo, ou palavra, como meio de condução das operações mentais.

- **Conceitos Cotidianos**: Aprendidos assistematicamente, estes conceitos dispensam a necessidade da escola para a sua formulação.

- **Conceitos Científicos**: Constituído por um sistema hierárquico de inter-relação, são os conceitos aprendidos na escola sistematicamente.

Nessa teoria há uma complexa relação entre o aprendizado e o desenvolvimento, ao contrário do que se tem em Piaget, onde o desenvolvimento antecede o aprendizado. Segundo Vygotsky, o aprendizado não coincide com o desenvolvimento, sendo que quando a criança aprende algum conceito, por exemplo: aritmética, o desenvolvimento dessa operação ou conceito apenas começou. Não há paralelismo entre aprendizagem e o desenvolvimento das funções psicológicas correspondentes. A aprendizagem dos alunos vai sendo assim construída mediante processo de relação do indivíduo com seu ambiente sócio-cultural e com o suporte de outros indivíduos mais experientes. É na zona de desenvolvimento proximal (ZDP) que a interferência desses outros indivíduos é mais transformadora. O conceito de ZDP é relativamente complexo, ele compreende a região de potencialidade para o aprendizado. No caso da criança, representa uma situação cognitiva em que ela só consegue resolver determinada tarefa psico-intelectual com auxílio de alguém mais experiente.

Para Vygotsky, como a aprendizagem impulsiona o desenvolvimento, a escola tem um papel essencial na construção do ser psicológico e racional. A escola deve dirigir o ensino não para etapas intelectuais já alcançadas, mas sim para estágios de desenvolvimento ainda não incorporados pelos alunos, funcionando como um incentivador de novas conquistas psicológicas. A escola tem ou deveria ter como ponto de partida o nível de desenvolvimento real da criança (em relação ao conteúdo) e como ponto de chegada os objetivos da aula que devem ser alcançados, ou seja chegar ao potencial da criança. Aqui o professor tem o papel explícito de interferir na zona de desenvolvimento proximal dos alunos, provocando avanços que não ocorreriam espontaneamente.

2.2.3 Divergências entre Piaget e Vygotsky

As teorias de Piaget e Vygotsky são teorias construtivistas e interacionistas, embora alguns autores considerem apenas Vygotsky com interacionista, o que parece não fazer sentido pois Piaget enfatiza em sua teoria a importância da interação com o ambiente, que pode ser física ou social, para que exista o processo de reconstrução das estruturas psicológicas do indivíduo. Veremos agora um resumo dos pontos divergentes entre as duas teorias. Mais detalhes em [Polônia (2000)] e [Franco (2001)].

Para Piaget a absorção direta de conhecimento se dá a partir da observação

e do contato com o meio externo. O meio exerce coerções e o indivíduo deve reconstruir-se para absorvê-las. Para Vygotsky o desenvolvimento da inteligência é produto da convivência com o outro. A intermediação no contato com o meio externo é sempre feita pelo outro e o indivíduo reelabora as informações.

Outros fatores importantes a serem considerados:

Segundo a teoria de Piaget o papel dos fatores internos do desenvolvimento preponderam sobre os externos (privilegia a maturação biológica); o desenvolvimento humano segue uma seqüência fixa e universal de estágios; a construção do conhecimento procede do individual ao social; a aprendizagem subordina-se ao desenvolvimento; o pensamento aparece antes da linguagem; a linguagem subordina-se aos processos de pensamento e ocorre após o alcance de determinados níveis de habilidades mentais.

Segundo a teoria de Vygotsky o papel dos fatores internos e externos do desenvolvimento variam conforme o ambiente (privilegia o ambiente social); não existe uma visão única do desenvolvimento humano; a construção do conhecimento procede do social para o individual; o desenvolvimento e a aprendizagem são processos interdependentes desde o nascimento; o pensamento é um processo interdependente da linguagem; a linguagem é a função central para o desenvolvimento cognitivo e é o que dá forma ao pensamento.

2.2.4 Considerações Finais

É importante esclarecer que não tem nenhum fundamento no construtivismo a prática pedagógica que se funda em "deixar que o aluno construa sozinho, sem interferência do professor", muito pelo contrário, como pode ser visto em [Franco (2001)], o objetivo do construtivismo é principalmente compreender melhor os processos que ocorrem no ato de ensinar/aprender. O construtivismo prega que devem haver professores capacitados, que conheçam as características psicológicas de cada faixa etária e que estimule o aprendiz a desenvolver seu potencial individualmente mudando a relação aluno professor para que este deixe de ser o "detentor da verdade"e passe a existir interação, permitindo que cada um desenvolva sua capacidade a seu tempo.

Também é um contra-senso ao construtivismo a prática de fazer com que os alunos se tornem alfabéticos até o final de um determinado bimestre, aprendam a somar até o final do outro, etc., como se fosse possível controlar o processo de aprendizagem desses alunos, padronizando suas estruturas cognitivas.

Outro ponto importante é como o raciocínio da criança é tratado. Um erro não deve ser tratado como um ato repreensível e definitivo. É importante observar o ra-

ciocínio que a criança desenvolveu a partir do conhecimento que já possui e levá-la a repensar sua resposta e direcioná-la corretamente. Em [Time (1999)], Seymour Papert relata uma interessante entrevista feita por Piaget a uma criança de 5 anos de idade chamada Julia:
Piaget: O que cria o vento?
Julia: As árvores.
Piaget: Como você sabe?
Julia: Eu vi elas abanando os braços.
Piaget: Como isso pode criar o vento?
Julia: Assim (abanando a mão em frente ao rosto de Piaget), só que elas são maiores, e existem muitas árvores.
Piaget: Então o que cria o vento no oceano?
Julia: Ele vai até lá da terra. Não são as ondas!

Sobre nenhum critério adulto as respostas de Julia estão corretas, mas elas estão erradas? Segundo Piaget não, elas são perfeitamente plausíveis com o conhecimento da criança e demonstram a capacidade de tirar conclusões a partir de observações e construir seu próprio conhecimento. À medida que se desenvolve e adquire mais informações ela será capaz de reformular seus conceitos e buscar novas respostas.

2.3 Seymour Papert e o Construcionismo

O matemático Seymour Papert nasceu na África do Sul, trabalhou como pesquisador na Universidade de Cambridge na Inglaterra entre 1954 e 1958, e entre 1958 e 1963 trabalhou com Jean Piaget no Centro de Epistemologia Genética da Universidade de Genebra, Suíça. Na década de 60 começou a trabalhar no MIT onde junto com Marvin Minsky fundou o laboratório de Inteligência Artificial. Hoje é reconhecido internacionalmente como um dos maiores pensadores sobre as formas nas quais os computadores podem mudar o processo de aprendizagem.

Durante o tempo em que conviveu com Piaget, Papert teve um contato intenso com as idéias do construtivismo, as quais foram a base de seus trabalhos. Ainda na década de 60, época em que os computadores custavam fortunas e pesavam centenas de quilos, Papert colaborou com a idéia pioneira de usar o computador para ajudar no processo de aprendizagem, em 1967 junto com uma equipe do Laboratório de Inteligência Artificial do MIT desenvolveu a linguagem LOGO, que inicialmente servia para controlar um pequeno robô.

A proposta de Papert é integrar as idéias do construtivismo com a tecnologia computacional, favorecendo assim a construção do conhecimento, pois o aluno torna-se ativo em sua aprendizagem, como pode ser visto em [Papert (1994)]:

> "A melhor aprendizagem ocorre quando o aprendiz assume o comando de seu próprio desenvolvimento em atividades que sejam significativas e lhe despertem o prazer"
>
> "A aprendizagem tem de fazer sentido para o aluno. O computador pode romper barreiras. O aluno que desenvolve seus programas ou acessa a rede, segundo seus interesses e necessidades e coloca a sua contribuição, sente-se fazendo parte da mudança e contribuindo para a sociedade. Para o aluno, o conhecimento necessário é aquele que lhe ajudará a obter mais conhecimento."

Papert denominou de **construcionista** esta abordagem pela qual o aprendiz constrói, por intermédio do computador, o seu próprio conhecimento como defendido em [Papert (1986)]. Ele usou esse termo para mostrar um outro nível de construção do conhecimento: a construção do conhecimento acontece quando o aluno constrói um objeto de seu interesse, como uma obra de arte, um relato de experiência ou um programa de computador. Segundo [Valente (2001)], na noção de construcionismo de Papert existem duas idéias não presentes no construtivismo de Piaget: primeiro, o aprendiz constrói alguma coisa ou seja, o aprendizado acontece por meio do fazer; segundo, o fato do aprendiz estar construindo algo do seu interesse e para o qual ele está bastante motivado, ou seja o envolvimento afetivo torna a aprendizagem mais significativa.

Em [Papert (1986)] é mostrado que, apesar de defender que o aluno deve construir seu conhecimento, o professor tem um papel fundamental, pois ele deve atuar como *facilitador de aprendizagem* e não apenas como uma pessoa que só transmite conteúdo, que muitas vezes não faz nenhum sentido para o aluno não interessado pelo assunto abordado.

> "Quando dizemos que nós educamos crianças, isto soa como algo que estamos fazendo para elas. Não é o que acontece. Nós não as educamos, nós criamos contextos para que elas aprendam."

Papert defende também a diversidade na educação, sendo um forte crítico ao sistema de ensino atual, que trata as crianças como se fossem iguais, gostassem das mesmas disciplinas, tivessem a mesma facilidade para aprender e raciocinar. Em [Papert (1999)] é citado:

> "A escola como conhecemos é baseada no modelo de *linha de montagem*. A linha de montagem era uma grande invenção quando Henry Ford a construiu. E a escola deve ter sido uma grande invenção quando foi feita, mas ela é um modelo de linha de montagem. Você entra na escola, você está na primeira série, no primeiro bimestre. Você faz o que o primeiro capítulo do livro texto diz. Você vai para o segundo bimestre, terceiro bimestre, a segunda série, terceira série. Isto é uma linha de montagem; em cada ponto algumas peças de conhecimento são instaladas."

Segundo Papert isto acontece porque os recursos tradicionais como o giz e o quadro negro são inflexíveis, e não há mais nenhuma razão para continuarmos presos apenas a isso, não há mais razão para segregar as crianças apenas por idade, enquanto poderíamos dividir as turmas de acordo com a maneira que cada um tem de fazer as coisas, com os interesses em comum.

Papert cita ainda:

> "Eu acho que cada criança no mundo é única e individual. (...) é maravilhoso como cada uma delas é diferente, fazem coisas de diferentes maneiras, pensam diferente. E então nós vamos levá-las à uma escola que vai homogeneizá-las?"

Capítulo 3

O Uso do Computador na Educação

3.1 Introdução

Neste capítulo são apresentadas uma visão de como os computadores têm sido usados nas escolas (especialmente as públicas), a necessidade e importância do uso de um *software* educacional e a importância da licença de um *software*. Também são discutidas possibilidades para que as escolas evitem o uso de *software* ilegal e economizem com a compra de licenças.

3.2 O Computador nas Escolas

Atualmente o computador é usado praticamente em todos os campos profissionais e ocupacionais, trazendo mais facilidade e eficiência para diversas tarefas. O sistema de ensino não pode ignorar esta ferramenta que pode se mostrar de grande valia para melhorar a forma como o conhecimento é disponibilizado ao aprendiz.

Tem-se visto com freqüência propagandas e campanhas do Governo para que as escolas tenham computadores, mas, de que adianta possuir computadores? Geralmente eles são mantidos longe do alcance dos alunos a maior parte do tempo. Além disso os professores não recebem uma preparação adequada para usá-los. Como apontado em [Valente (2001)], o computador tem sido usado para informatizar o sistema de ensino como já existia antes. O aluno aprende a operar os aplicativos mais difundidos no mercado e às vezes usa programas que permitem que ele responda a questionários ou acesse grandes bases de dados. Aprender a

operar um computador ou um determinado *software* é válido, mas não prepara o aprendiz para raciocinar ou refletir, tampouco foge do antigo sistema de ensino. A promessa de que o computador pode revolucionar o ensino não deve consistir em *ensinar a usar o computador* e sim *usar o computador para ensinar*.

Existe atualmente por parte do Governo e mesmo da sociedade apenas a preocupação quantitativa sobre este assunto. É conveniente para o Governo, divulgar que um grande percentual das escolas públicas brasileiras possuem essas *maravilhas tecnológicas*, que parecem ser coisas de outro mundo para a maior parte da população. Mas o sentido desta preocupação deveria ser a criação um sistema de ensino onde realmente o computador seja usado para ajudar o aluno a aprender. Seymour Papert em [Papert (1999)] faz uma analogia entre o uso do computador na escola e a invenção da escrita:

> "Imagine uma sociedade na qual a escrita não foi inventada ainda, então não há livros nem lápis. As pessoas ensinam verbalmente e o aprendizado se dá pelo ato de ouvir.
>
> Um dia alguém inventa a escrita e o lápis. Alguém diz: "Uau! Isto pode ser maravilhoso para a educação, pode revolucionar o ensino. Então, vamos por um lápis em cada sala de aula no país e ver o que acontece."Bem, o lápis não pode fazer nada, pode?"

Da mesma forma se colocarmos um computador em cada escola ele não pode fazer nada para melhorar, muito menos revolucionar o ensino e, provavelmente só será usado por alguma secretária para jogar um "joguinho de cartas".

3.3 O *Software* Educacional

Como acontece em todas as áreas onde o computador é usado, ele só é realmente útil com um *software* adequado. Percebendo isso, várias universidades e empresas desenvolvem aplicativos educacionais, que visam melhorar e facilitar a tarefa de professores e alunos. Este tipo de *software* no mínimo torna a aprendizagem mais atraente ao aluno, além das eventuais melhorias propostas, o que com certeza melhora o rendimento do aprendiz.

Atualmente podem ser encontrados programas de computador que focam o aprendizado de alunos desde a escola primária, até o ensino médio e superior. A Internet tem grande importância para divulgação e distribuição destes programas. Porém, antes de se adotar um *software*, é importante que este seja avaliado criteriosamente, pois como mostra [Chaves (1999)], a maioria deles é feita por *amadores*

em educação, sem nenhum embasamento teórico pedagógico. Em [Souza (1998)] é citado que um *software* educacional deve ser avaliado pelas seguintes características:

- Atende as necessidades do objetivo curricular?
- Tem relevância pedagógica e os objetivos são claros?
- É de fácil uso pelos alunos?
- Permite modificações a fim de atender às necessidades individuais do aluno?
- Pode ser utilizado em várias situações de sala de aula (individual, pequenos ou grandes grupos)?
- Passa por várias formas de aprendizagem (visual, auditiva, numérica, verbal)?
- Como são tratados os erros dos alunos?
- Qual o controle do aluno sobre o *software*?
- Existe um bom manual para o professor e para o aluno?
- Os recursos computacionais são utilizados adequadamente?

A partir deste questionamento pode-se chegar a bons programas que podem ser usados em determinados conteúdos, inclusive alguns que não são explicitamente da área educativa, como jogos, que podem ser interessantes em algumas disciplinas, como língua estrangeira e matemática entre outros.

3.4 O Problema do Software Não Licenciado

Com freqüência empresas desenvolvedoras de *software* para a área da educação procuram as escolas afim de divulgar e comercializar seus produtos, oferecendo inclusive treinamento ao professor, o que é muito importante, mas normalmente, apenas escolas particulares podem pagar por eles.

Existem aplicativos educacionais gratuitos, apesar de normalmente *não baterem à porta* podem ser encontrados na Internet sem maiores dificuldades. Surge então outro problema, a maioria absoluta destes aplicativos são para o sistema *Windows*, e este sistema não é gratuito, pelo contrário, tem um custo alto, como

Windows XP Professional	US$ 299,00
Windows XP Professional (Licença Adicional)	US$ 269,00
Windows XP Home Edition	US$ 199,00
Windows XP Home Edition (Licença Adicional)	US$ 189,00
Office XP Full Standard	US$ 149,00

$Fonte: www.microsoft.com (11/03/2002)$

Tabela 3.1: Preços dos Principais Sistemas da $Microsoft$

pode ser visto na tabela 3.1 (Existem pacotes especiais para escolas e universidades e dependendo do número de licenças o desconto pode ser de até 50%).

Alguém pode se indagar: "Mas a maioria das escolas não usam este sistema?", a resposta é sim. Existem vantagens em se adotar o sistema *Windows* em uma escola, primeiro pela compatibilidade com a maioria dos aplicativos desenvolvidos para computadores pessoais. Segundo, porque muitas pessoas têm noções básicas de operação deste sistema. Mas em escolas públicas onde se promovem campanhas para arrecadar dinheiro e comprar computadores será que pagaram pela licença do *Windows*? Será que a empresa da qual compraram o computador informou da necessidade de se pagar pelo sistema operacional, editor de texto e planilha eletrônica? Muitas pessoas não sabem da existência e da necessidade da licença de um *software*, o que pensam intuitivamente é que aquilo faz parte do computador. O que vendedores de *hardware* corriqueiramente fazem, para reduzir o preço do produto, é instalar um sistema *pirata*, ou seja, com uma licença inválida, logo ilegal. Estas empresas confiam na debilidade da fiscalização que ocorre no Brasil para cometerem esta prática e saírem impunes.

Parece um tanto contraditório que escolas que deveriam educar e conscientizar a sociedade, cometam uma prática ilegal, e muitas vezes sem consciência do que estão fazendo.

3.5 Licença GPL

Uma solução para contornar o problema do alto investimento necessário para licenciar um *software*, é a adoção de um *software* gratuito, como os distribuídos sob a licença GPL. GPL ou **General Public Licence**, é uma licença criada pela GNU, que é uma Organização não governamental empenhada em criar, distribuir e proteger o *software* gratuito de código aberto. Um programa distribuído sob esta licença pode ser investigado, alterado, redistribuído e instalado em quantas máquinas se

desejar. A restrição básica da licença é que ele não pode se tornar um programa comercial, ou seja não pode ser vendido, mesmo que tenha sido melhorado. Como pode ser visto em [GNU (2001)]:

> "As licenças de muitos *software* são desenvolvidas para restringir sua liberdade de compartilhá-lo e mudá-lo. Contrária a isso, a Licença Pública Geral GNU (GPL) pretende garantir sua liberdade de compartilhar e alterar *software* livres, garantindo que o *software* será livre e gratuito para os seus usuários"

3.6 O Sistema *Linux*

A maior parte das pessoas pensam na adoção do sistema *Windows* como algo imperativo. Isso ocorre principalmente pelo desconhecimento da existência e vantagens de outros sistemas operacionais. Um dos sistemas que substitui com vantagens o *Windows* é o *Linux*, o qual é distribuído sob licença GPL.

Como pode ser visto em [Empório Linux (2002)], *Linux* é um sistema operacional criado inicialmente pelo finlandês Linus Torvalds em 1991, sem grandes perspectivas, apenas como um projeto pessoal. Seu objetivo era criar um sistema baseado em Unix que pudesse ser usado em seu micro computador. Quando anunciou os primeiros resultados na Internet, várias pessoas se interessaram e o *Linux* começou a se difundir pelo mundo. O projeto cresceu e outros programadores começaram a colaborar espontaneamente, sem objetivar lucros. Atualmente, a lista de colaboradores do kernel *Linux* inclui mais de 250 pessoas do mundo inteiro, além de milhares que contribuem com vários aplicativos.

Várias instituições no mundo inteiro vêm adotando o *Linux*, não só pelo fato de ser gratuito, mas pelas diversas vantagens que apresenta sobre o *Windows*. No Brasil podemos citar o exemplo do estado do Rio Grande do Sul, que adotou o *Linux* em todas as instituições públicas estaduais.

Uma pessoa que sabe operar o *Windows* não terá dificuldades de assimilar o *Linux*, e para quem vai aprender a operar um computador, o *Linux* pode inclusive ser mais fácil, prático e seguro, pois o *Linux* não permite que um usuário comum desconfigure ou danifique todo o sistema, como acontece nas versões domésticas do *Windows*.

3.6.1 Distribuições *Linux*

Existem hoje pelo mundo várias *distribuições Linux*, que são pacotes contendo o *Kernel* e vários aplicativos, como editores de texto, compiladores, visualizadores de imagens, tocadores de mp3, jogos, entre outros. Além disso, cada distribuição tem seu próprio aplicativo de instalação.

As *distribuições* mais conhecidas e difundidas no Brasil são:

- **Red Hat** - É a mais popular distribuição de *Linux* tanto entre usuários como em desenvolvedores. Sua instalação é fácil e rápida.
- **Conectiva** - Quase totalmente traduzida para o português e conta com uma boa documentação, essa distribuição baseada no Red Hat, é a única genuinamente brasileira.
- **Debian** - Totalmente desenvolvida e mantida através do trabalho de voluntários, no espírito do *Linux* e do projeto GNU, é a distribuição mais fácil de se atualizar e manter atualizada.
- **Corel** - Baseada na Debian. Destaca-se pela sua facilidade de instalação e configuração.
- **Mandrake** - Tem como destaque o seu instalador e configurador DrakX, que permite a instalação de acordo com o grau de conhecimento do usuário.
- **SuSE** - É a mais famosa distribuição na Alemanha e na Europa. A instalação é fácil, o que a torna atraente para leigos.

3.6.2 Principais Características

Além de ser um *software* livre, o *Linux* possui outras importantes características que o destacam frente a outros sistemas operacionais, como pode ser visto em [Empório Linux (2002)]:

- **Multitarefa Real**: É possível executar vários programas ao mesmo tempo.
- **Multi-Usuário**: Vários usuários podem usar a mesma máquina, local ou remotamente. O *Linux* fornece mecanismos de acesso que permitem proteger os arquivos, por exemplo, é possível especificar quais usuários podem ler, alterar, apagar ou executar um aplicativo, pasta ou documento.

- **Estabilidade**: Os recursos do kernel *Linux* são exaustivamente testados por milhares de pessoas antes de serem lançados oficialmente, resultando num número reduzido de *bugs*. Isso é uma importante característica do modelo de desenvolvimento do *Linux*, e só é possível porque o *Linux* é *software* de código aberto. No lado técnico, um exemplo de estabilidade muito citado é o fato do *Linux* não travar. Como ele roda sempre em modo protegido, se um programa trava, o resto do sistema não é afetado, ao contrário do *Windows*.

- **Fortes Recursos para Rede**: O *Linux*, sendo derivado do Unix, tem suporte nativo à rede. Evidentemente, essa familiaridade com redes se estende à Internet. O *Linux* possui os *drivers* mais rápidos e suporta os principais protocolos.

Em todos estes aspectos o *Linux* é visivelmente superior ao *Windows*. Além disso, outro fator importante é que vários aplicativos existentes no *Linux* suportam formatos de aplicativos do *Windows*, como por exemplo arquivos *.doc*, o contrário não acontece. O *Linux* consegue ler partições no formato *Windows* e em mais de 30 sistemas de arquivo diferentes, contra os 3 do *Windows*, que não consegue ler partições *Linux*.

3.6.3 Deficiências do Sistema *Linux*

Segundo [Empório Linux (2002)], existem alguns fatores que aumentam a resistência à adoção do sistema *Linux* entre elas destacam-se:

- **Instalação**: Apesar da grande evolução que já ocorreu, o processo de instalação do *Linux* não é simples como o do *Windows*. O *Linux* necessita de partições no disco com diferentes tipos de sistemas de arquivos, e que exigem conhecimentos mais avançados por parte de quem o instala.

- **Configuração**: A configuração do *Linux* é, em sua maioria, feita a partir de arquivos texto, isto traz algumas vantagens, mas tornam a configuração difícil para um usuário comum. A maioria das distribuições já trazem ferramentas gráficas que permitem fazer todas as configurações básicas permitidas ao usuário, mas configurações avançadas exigem um grau de conhecimento maior por parte de quem a executa.

- **Suporte Técnico**: Não existem ainda muitas empresas preparadas para oferecer suporte técnico para *Linux*, principalmente em pequenas e médias cidades. Há também a falta de suporte por parte dos fabricantes de hardware que

muitas vezes não disponibilizam drivers para *Linux*, ou mesmo as especificações para que estes drivers sejam implementados.

- **Falta de Aplicativos**: Desenvolver sistemas para *Linux* ainda não é tão simples como para *Windows*, por isso apenas desenvolvedores com conhecimento mais avançado conseguem criar aplicativos em *Linux*. Apesar de terem uma qualidade melhor, eles não se equiparam ainda em quantidade e diversidade, aos existentes para o ambiente *Windows*.

Estas deficiências existem, mas não são críticas e tampouco difíceis de serem contornadas. O próprio dinamismo que existe na evolução do *Linux* faz com que rapidamente mais e mais soluções apareçam. O *Linux* vem conseguindo cada vez mais adeptos em todo o mundo, e junto com sua popularização vem surgindo empresas especializadas e aplicativos para as mais diversas áreas e aplicações.

3.7 Comentários Finais

Não é incumbência deste projeto avaliar qual sistema é mais adequado ou mais viável para as escolas, o que este propõe é o desenvolvimento de um *software* educacional de qualidade, que possa ser usado nos dois sistemas, oferendo assim ao usuário a **liberdade** de analisar e escolher o sistema que deseja utilizar, contanto que respeite as licenças desses sistemas.

É opinião pessoal do autor que o sistema *Linux* é mais adequado e viável para as escolas, sendo que resolveria vários problemas freqüentes nas escolas, como: infecção por vírus de computador; danificação do sistema por usuários mal intencionados ou despreparados; falta de privacidade para os usuários; falta de registro e controle sobre as ações dos usuários; falta de dinheiro para compra de licenças.

Capítulo 4

A Linguagem LOGO

4.1 Introdução

Nos capítulos anteriores foram discutidos o construtivismo e o uso dos computadores na educação. O objetivo deste capítulo é apresentar a linguagem de programação LOGO, que integra estas duas áreas. Serão apresentadas suas principais características, vantagens, aplicações e os comandos e sintaxe.

4.2 O que é LOGO

LOGO é uma linguagem de programação voltada para o ambiente educacional, se fundamenta na filosofia construtivista e em pesquisas na área de Inteligência Artificial. Esta linguagem é usada para comandar um cursor, normalmente exibido como uma tartaruga, com o propósito de ensinar ao cursor novos procedimentos além dos que ele já conhece, afim de criar desenhos ou programas. O grau de sofisticação desses desenhos ou programas depende do nível do usuário, que pode ser tanto uma criança de 8 anos como um adulto, e pode ensinar ao cursor como desenhar um simples quadrado ou como plotar um gráfico complexo.

Segundo [Santos (2000)] o nome "Logo"foi uma referência a um termo grego que significa: *pensamento, raciocínio, discurso*, ou ainda, *razão, cálculo, linguagem*.

A linguagem LOGO foi desenvolvida na década de 60 no MIT - *Massachussets Institute of Technology*, Cambridge, Massachusetts, Estados Unidos - por uma equipe chefiada por Seymour Papert. Em meados da década de 70 começou a ser testada fora dos laboratórios, e hoje é difundida em todo o mundo, e apontada

por especialistas em educação como um dos melhores e mais importantes *software* educacional, como pode ser visto em [Ramos (1996)].

4.2.1 A Metáfora da Tartaruga

A grande maioria dos interpretadores da linguagem LOGO, usam como cursor a imagem de uma tartaruga. Na verdade esta metáfora surgiu de uma outra, quando a linguagem LOGO foi criada, sua finalidade era controlar um pequeno robô que riscava o chão por onde passava. A semelhança do robô com uma tartaruga logo chamou a atenção e ele passou a ser chamado assim. Com a migração para o ambiente virtual a metáfora da tartaruga foi mantida e se tornou o símbolo da linguagem.

4.2.2 A Filosofia LOGO

Como apontado por [Santos (2000)], o LOGO propõe uma metodologia de ensino que busca, através de uma linguagem semelhante à natural, facilitar a comunicação entre o usuário e o computador, e proporcionar a criação de modelos através de formas geométricas e do raciocínio lógico. Propõe também, que o aluno seja ativo construtor de seus próprios conhecimentos, desenvolvendo assim sua capacidade intelectual. O professor deve permitir a reflexão do aluno, ao contrário do modelo tradicional onde reina o autoritarismo. O aluno, através do erro, é condicionado a refletir novas formas de resolução do problema, ou seja, ele tem a chance de aprender com seus próprios erros e é estimulado a tentar.

> "Quando acontece um erro, este torna-se um objeto de análise para que seja identificado e reformulado, desencadeando aprendizagem e desenvolvimento. Esse processo estabelece um ciclo de descrição-depuração - reflexão-depuração, que foi implantado na programação de computadores."[Papert (1994)]

No LOGO considera-se o erro como um importante fator de aprendizagem, o que oferece oportunidades para que o aluno entenda porque errou e busque uma nova solução para o problema, investigando, explorando, descobrindo por si próprio, ou seja, a aprendizagem pela descoberta.

> "O Logo propõe um ambiente de aprendizagem no qual o conhecimento não é meramente passado para o aluno, mas, uma forma de trabalho onde esse aluno em interação com os objetos desse ambiente, possa desenvolver outros conhecimentos, por exemplo: conceitos

geométricos ou matemáticos. Propicia ao aluno a possibilidade de aprender fazendo, ou seja, ensinando a tartaruga a resolver um problema, seguindo a linguagem de programação. O aluno pode, ao ver o resultado da execução, comparar suas expectativas originais com o produto obtido, analisando suas idéias e os conceitos que usou. Se houver um erro o aluno pode depurar o programa e identificar a origem do erro, usando o erro de modo produtivo, para entender melhor suas ações."[Zacharias (1998)]

4.2.3 Principais Características

Segundo [Correia et alii (1999)], as principais características da linguagem LOGO são:

- **Amigabilidade**: É uma linguagem de fácil aprendizado e uso.
- **Modularidade e Extensibilidade**: É possível criar novos comandos para a linguagem, usando a própria linguagem LOGO. Por exemplo podemos criar um comando *quadrado*, que desenha automaticamente um quadrado, ao invés de desenharmos cada um dos lados.
- **Interatividade**: Oferece uma resposta imediata e mensagens informativas sobre o comando aplicado.
- **Flexibilidade**: LOGO pode ser usado com crianças no ensino fundamental ou alunos de curso superior.
- **Capacidade**: É uma linguagem de programação poderosa, possuindo ferramentas necessárias para criar programas com diversos graus de sofisticação.

4.2.4 O Ambiente Logo

Nos primórdios de seu desenvolvimento a linguagem LOGO comandava um robô ligado a um *mainframe*. Robôs são caros e delicados, por isso no fim da década de 70 surgiu o *software* que interpretava a linguagem LOGO, e comandava uma *tartaruga* virtual, na tela do computador. Assim são a maioria dos interpretadores LOGO nos dias de hoje. Normalmente possuem uma grande área de desenho onde a tartaruga pode percorrer e desenhar; uma caixa de texto na parte inferior da tela onde são digitados os comandos; um repositório, que é uma lista onde são guardados os comandos executados e as mensagens de erro ou informativas. O menu

permite configurar linhas e fontes e abrir um editor de novos comandos, que são chamados procedimentos. A interface dos interpretadores LOGO se assemelham a um jogo de computador, o que desperta a curiosidade e estimula mais a criança a explorá-lo.

4.3 O LOGO na Educação

4.3.1 Porque Usar LOGO?

A linguagem LOGO possui um forte embasamento teórico, e foi desenvolvida por uma equipe respeitada em todo o mundo. Além disso é uma ferramenta que permite a adoção da filosofia construtivista, de uma maneira relativamente fácil. Em [FAC (2002)] são citadas outras razões para se adotar o LOGO:

- É um *software* clássico que nunca se esgota e pode ser usado em todas as idades.
- Estimula a resolução problemas e é um instrumento para a exploração de idéias.
- O LOGO ajuda a compreender conceitos lógicos e matemáticos importantes como aritmética e geometria.
- O aprendizado é ativo, explorativo e vivencial.
- Ao observar os programas dos estudantes é possível encontrar formas de compreender como estão trabalhado suas mentes no processo de resolução dos programas, além das estratégias e estilos que usaram.
- A forma de usar o LOGO e outros aplicativos não se contrapõem, se complementam.
- O LOGO abrange várias áreas, não uma específica.

Ocorre também que o LOGO é um *software* muito difundido, por isso é fácil encontrar manuais, tutoriais, fóruns e exemplos através da Internet ou livros e apostilas em uma biblioteca. Além de versões gratuitas desenvolvidas por universidades ou organizações.

4.3.2 O Papel do Professor

A princípio não houve preocupação com o papel do professor no ambiente LOGO. Em [Papert (1980)] Papert deixa transparecer que *idéias poderosas* surgiriam espontaneamente da atividade do aluno ao programar em Logo e isso aconteceria sem uma maior intervenção do professor, cabendo-lhe apenas auxiliar os alunos no que diz respeito à sintaxe do Logo. Como decorrência disso, logo surgiu um grande descontentamento com os resultados obtidos, já que estes deixaram muito a desejar em relação ao que se anunciava que o Logo poderia fazer pela educação.

> "Hoje sabemos que o papel do professor no ambiente Logo é fundamental, que o preparo do professor não é trivial não acontecendo do dia para a noite."[Valente (1996)]

Atualmente, Papert também concebe o papel do professor sob uma ótica bastante diferente. Em uma de suas obras mais recentes, [Papert (1994)], assume que durante muito tempo subestimou o professor no que diz respeito a sua função no ambiente Logo.

Para que toda a potencialidade do LOGO seja alcançada o professor deve saber não apenas operar o computador, mas dominar a linguagem e ser capaz de criar novas maneiras de apresentar conceitos que podem ser complicados para uma criança, como por exemplo a medida de um ângulo em graus ou da unidade *pixel*. O professor deve ser um facilitador, ajudando a desenvolver a autonomia do aluno e incentivando suas habilidades individuais.

4.3.3 Áreas de Aplicação

O LOGO pode ser facilmente aplicado em várias áreas da matemática, como:

- Geometria
- Trigonometria
- Álgebra
- Funções
- Estatística

Porém os *software* LOGO atuais têm implementado cada vez mais comandos que permitem uma maior interação com recursos multimídia e dispositivos ligados às saídas do computador. Existem projetos que utilizam LOGO nas mais diversas áreas como:

- Física
- Música
- Robótica
- Biologia
- Computação Gráfica
- Lingüística
- Simulação de Sistemas Complexos

4.4 Características da Linguagem

A linguagem LOGO é parecida com a língua nativa dos alunos que vão utilizá-la, existem versões padronizadas em várias línguas. Como seus comandos são baseados na linguagem cotidiana, e não em uma linguagem técnica, pessoas *comuns* e principalmente crianças tem muita facilidade para assimilar não só os comandos, mas também os mnemônicos associados a alguns deles, por exemplo, em inglês o comando forward faz a tartaruga andar para frente e seu mnemônico é fw, em português este comando é o parafrente e o mnemônico é pf.

Normalmente o interpretador não diferencia maiúsculas de minúsculas e exige que os comandos cujos nomes contém acentos, sejam acentuados corretamente, por exemplo *potência* e *paratrás*. LOGO possui também comandos para o tratamento de listas, herança da linguagem LISP, da qual foi derivada. LISP é muito usada em sistemas na área de inteligência artificial, como o LOGO foi desenvolvido no laboratório de inteligência artificial do MIT, sofreu influências desta linguagem. Este conceito só é trabalhado com usuários mais experientes, para indicar uma lista são usados colchetes "[]".

Diferente da maioria das linguagens uma *string* (cadeia de caracteres) não é delimitada por aspas, *string* em LOGO possui apenas uma seqüência de caracteres sem espaços ou seja uma palavra não composta, para indicá-la como *string* uma única aspa dupla deve ser colocada no início da palavra mas não pode ser fechada. Para usar seqüências com várias palavras deve-se usar uma lista. O único separador de comandos e parâmetros é o espaço, não existem caracteres para indicar fim de linha ou delimitar parâmetros. Para inserir comentários nos procedimentos é usado um ; (ponto e vírgula), que faz com que o resto da linha a partir dele seja desprezada.

4.4.1 Comandos Básicos

Para começar a programar em LOGO é essencial aprender a movimentar a tartaruga pela tela. Na tabela 4.1 são apresentados os comandos básicos usados para este fim, onde x representa um valor numérico, que pode também ser uma variável ou uma função que retorne um valor numérico. Cada passo corresponde a 1 pixel (menor ponto mensurável da tela). É importante lembrar que a tartaruga possui um *"lápis"* no *"umbigo"*, o qual risca o caminho por onde passa, existem comandos para manipular este lápis, apresentados na tabela 4.2.

Comando	*Mnemônico*	*Ação da Tartaruga*
parafrente x	pf	Anda x passos para frente
paratrás x	pt	Anda x passos para trás
paradireita x	pd	Gira x graus para a direita
paraesquerda x	pe	Gira x graus para a esquerda
paracentro	pc	Volta para a posição original
pare		Pára a execução dos procedimentos, inclusive o $repita$

Tabela 4.1: Comandos Básicos Para Movimentação da Tartaruga

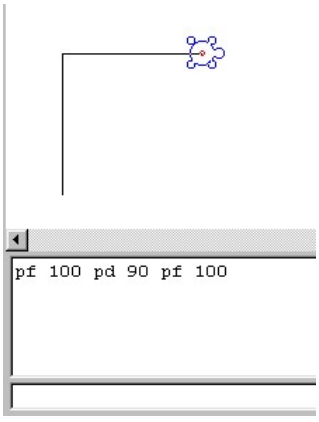

Figura 4.1: Exemplo de Movimentação da Tartaruga

Comando	*Mnemônico*	*Ação da Tartaruga*
uselápis	ul	Risca a tela por onde passa
usenada	un	Anda sem riscar a tela
useborracha	ub	Apaga os riscos por onde passa

Tabela 4.2: Comandos Para Manipulação do Lápis

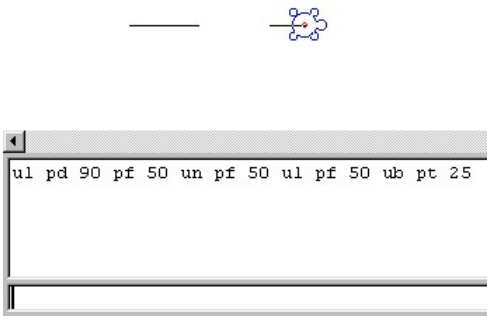

Figura 4.2: Exemplo de Manipulação do Lápis

4.4.2 Comandos Importantes

Além dos comandos básicos para controlar a tartaruga, existem vários comandos que facilitam e auxiliam a criação de programas.

Comandos de Desenho

Existem comandos definidos no LOGO para desenhar formas que não são tão fáceis de serem criadas como um quadrado ou triângulo, esses comandos são mostrados na tabela 4.3:

Comando	*Descrição*
circunferência x	Desenha uma circunferência de raio x
arco α x	Desenha um arco de α graus com raio x
elipse x y	Desenha uma elipse de raios x e y
rotule [$texto$]	Imprime $texto$ na tela na direção lateral da tartaruga

Tabela 4.3: Comandos de Desenhos

Comandos de Movimentação por Coordenadas

Existem comandos para manipular a tartaruga através de coordenadas cartesianas, apesar de não serem tão simples de entender, principalmente para uma criança, estes comandos são de grande utilidade em alguns programas.

Comando	*Mnemônico*	*Ação da Tartaruga*
mudex x		Muda a coordenada x da tartaruga
mudey y		Muda a coordenada y da tartaruga
mudeposição [x y]	mudepos	Muda as coordenadas x e y da tartaruga

Tabela 4.4: Comandos de Movimentação por Coordenadas

Comandos de Consulta

Para consultar o valor de uma variável ou expressão existem comandos de consulta, os quais apenas imprimem o valor retornado na janela de comandos. Estes comandos são muito úteis para fazer cálculos afim de usar o resultado para levar a tartaruga ao lugar certo. Estes comandos são: **escreva** ou **esc** e **mostre**. A diferença entre eles é referente ao tratamento de listas e é mostrada na tabela 4.5.

Comando	*Resultado*
escreva [a b c d]	a b c d
mostre [a b c d]	[a b c d]
escreva 8+8	16
mostre 8+8	16
escreva 2=3	falso
mostre 2=3	falso
escreva 2+5>=10-3	verd
mostre 2+5>=10-3	verd

Tabela 4.5: Comandos de Consulta

4.4.3 Estrutura de Repetição

Às vezes para desenhar formas geométricas é necessário repetir várias vezes a mesma seqüência de comandos, por exemplo para fazer um quadrado de lado 50 é preciso digitar: pf 50 pd 90 pf 50 pd 90 pf 50 pd 90 pf 50 pd 90. Para que não

seja necessário repetir tantas vezes os mesmos comandos, existe uma estrutura que faz este trabalho automaticamente, o **repita**, sua sintaxe é: **repita** x **[**$comandos$**]**, onde x é o número de vezes que os $comandos$ dentro da lista serão repetidos. Exemplo: repita 4 [pf 50 pd 90] veja o resultado na figura 4.3.

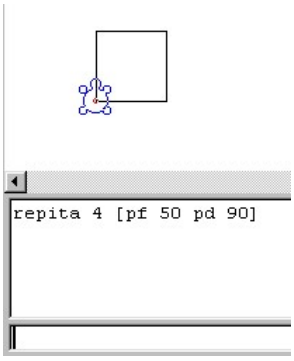

Figura 4.3: Exemplo da Estrutura Repita

4.4.4 Estrutura Condicional

Em toda linguagem de programação estruturada existe uma estrutura condicional, o LOGO possui a estrutura **se**, que avalia uma condição e executa ou não uma lista de comandos. Sintaxe: **se** $predicado$ **[**$comandos1$**] [**$comandos2$**]**, onde $predicado$ deve ser uma expressão booleana (retorna verdadeiro ou falso), caso seja verdadeira $comandos1$ será executado, senão, $comandos2$ será executado, caso tenha sido especificado, apenas a primeira lista de comandos é obrigatória. Exemplo: se 2+3>5 [pf 100 pd 135][pt 100 pe 45]. Veja outro exemplo na figura 4.4.

4.4.5 Variáveis

Na linguagem LOGO não é necessário declarar as variáveis. Também não existem tipos definidos de dados, ou seja é possível atribuir qualquer valor a qualquer variável. Uma variável é criada quando se atribui um valor a ela, caso ela já exista o valor é sobreposto. Para atribuir valores a uma variável é preciso usar o comando **atribua** ou **atr**. Sintaxe: **atribua** $nomevalor$, onde nome deve ser uma palavra sem espaços e, para indicar que é uma *string*, deve ser iniciada com uma "(aspa

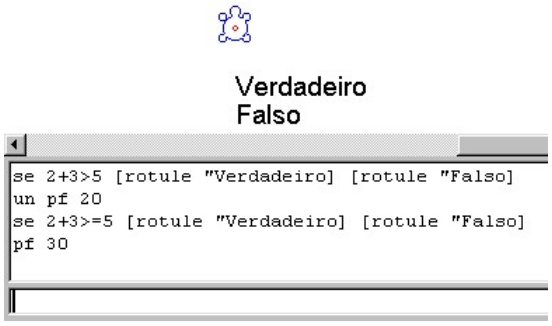

Figura 4.4: Exemplo da Estrutura Se

dupla).
Exemplo: atribua "x 0 ou ainda atribua "x 2+2
É importante lembrar que o nome da variável não vai conter a aspa. Para acessar e usar o valor da variável é preciso colocar um : (dois pontos) e depois o nome da variável.
Exemplo: atr "x 100 atr "y 90 pd :y pf :x (resultado na figura 4.5)

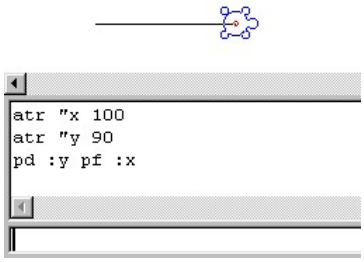

Figura 4.5: Exemplo do Uso de Variáveis

4.4.6 Funções de Manipulação de Listas

A linguagem LOGO assim como a LISP, possui funções para tratamento e manipulação de listas. Estas funções geralmente são usadas por usuários mais avançados. Na tabela 4.6 são mostrados exemplos do uso destas funções.

Função	*Mnemônico*	*Exemplo*	*Resultado*
primeiro	pri	esc pri [3 6 9 12]	3
último	ult	esc ult [3 6 9 12]	12
semprimeiro	sp	esc sp [3 6 9 12]	[6 9 12]
semúltimo	su	esc su [3 6 9 12]	[3 6 9]
juntenoinício	ji	esc ji 5 [3 6 9 12]	[5 3 6 9 12]
juntenofim	jf	esc jf 15 [3 6 9 12]	[3 6 9 12 15]

Tabela 4.6: Comandos de Tratamento de Listas

4.4.7 Criação de Novos Procedimentos

Uma das mais importantes características do LOGO é ser uma linguagem extensível, ou seja, novos comandos podem ser *ensinados* à tartaruga. Estes comandos, ou como mais comumente chamados procedimentos, são criados usando a própria linguagem LOGO, e devem ser salvos em um projeto para que possam ser usados posteriormente. Para ensinar a tartaruga um novo procedimento é usado o comando **aprenda**, devem ser passados o nome do novo procedimento e os parâmetros que ele irá possuir, então cada interpretador possui uma forma particular de editor para que os comandos que definem o procedimento sejam inseridos, ao final dos comandos a instrução **fim** deve ser usada para fechar o procedimento. Geralmente existem no menu opções para criar, editar ou excluir procedimentos.

Capítulo 5

O wxLogo

5.1 Introdução

O wxLogo é um interpretador multi-plataforma para a linguagem LOGO, que foi proposto e desenvolvido durante este trabalho. Já foi testado nos ambientes *Linux* e *Windows*, e com possibilidade de trabalhar em vários outros sistemas operacionais. Neste capítulo são mostradas suas novidades em relação a outros interpretadores, os recursos já disponíveis, a interface e a estrutura dos arquivos de ajuda.

5.2 A Origem

O nome wxLogo partiu da idéia de usar as letras que normalmente representam os sistemas operacionais *Windows* (W) e *Linux* (X) junto com o nome da linguagem interpretada. É feita uma alusão à frase: *"Windows and Linux's LOGO"*.

A idéia inicial do desenvolvimento do wxLogo surgiu do Prof. Joaquim Uchôa, o qual é professor de graduação no curso de Ciência da Computação, e no curso de pós-graduação *lato sensu: Informática em Educação*, da Universidade Federal de Lavras. Ao constatar a dificuldade de se encontrar um interpretador LOGO para o ambiente *Linux*, principalmente em português, ele sugeriu o início do desenvolvimento da interface multi-plataforma, e alguns comandos básicos, como projeto para a disciplina de computação gráfica, ministrada pelo Prof. Bruno Schneider. O primeiro protótipo foi desenvolvido pelos alunos Eduardo Teodoro S. Júnior e Marcos A. Domingues. Na continuação deste projeto a interface foi reestruturada e todas as funções de análise e interpretação de código substituídas, a fim de se criar funções que melhor aplicassem as teorias e formalismos das disciplinas de

linguagens formais e autônomos e compiladores, gerando assim um interpretador mais robusto e flexível.

5.3 As Novidades

Para definir como seria a interface e quais comandos deveriam ser mantidos ou melhorados no wxLogo, foram consultados os pós-graduandos do curso de *Informática em Educação*, dos quais vários têm experiência com a aplicação da linguagem, e os professores do curso, além opinião do autor. A seguir são listadas as mudanças mais significativas em relação aos interpretadores em português adotados no curso, que são o *Super Logo 3.0* e o *SLogo95*, disponíveis gratuitamente em [Nied (2001)], e que serviram de referência para o wxLogo.

5.3.1 Interface

A interface do wxLogo objetivou ser mais simples, objetiva e atrativa que as demais avaliadas, as principais mudanças são:

- A interface que era constituída de duas janelas independentes, uma com a área onde a tartaruga andava e outra com os botões e caixas de texto para entrada e armazenamento dos comandos foi transformada em uma única janela, na qual a área útil da tela é maior.

- A tartaruga passou a ser desenhada com linhas coloridas ao invés de apenas pretas se tornando mais chamativa.

- A direção inicial da tartaruga (apontando para cima) que normalmente é considerada a direção 0, passou a ser 90, o 0 passou a ser a posição *3:00* horas do relógio, como no círculo trigonométrico adotado na matemática.

- Existem apenas 3 botões na interface, um para limpar a tela, outro para limpar a janela de comandos já digitados e outro para executar os comandos da caixa de entrada, ao contrário dos outros interpretadores que possuíam vários botões, a maioria pouco usados.

- Foram inseridos abaixo dos botões, em uma área normalmente ociosa, as coordenadas da tartaruga e o ângulo que indica a direção em que ela se encontra.

- Ao se clicar na janela de comandos, o texto da linha clicada é copiado para a caixa de entrada, como nas outras versões, a diferença no wxLogo é a opção de selecionar uma parte do texto contendo uma ou várias linhas sem que o conteúdo da caixa de entradas seja alterado.

5.3.2 Procedimentos do Usuário

As novidades na forma como o wxLogo trata os procedimentos salvos em arquivo, são no sentido de informar melhor ao usuário onde e o que ocorreu durante a interpretação do arquivo, além de evitar a desorganização dos procedimentos.

- Para cada procedimento do arquivo carregado com sucesso é exibida uma mensagem confirmando o nome do procedimento e o êxito da operação.
- Sempre que um arquivo for aberto todos procedimentos que eventualmente estiverem abertos são fechados, se não estão salvos, uma mensagem adverte e permite esta opção. Os outros interpretadores simplesmente acrescentam os novos procedimentos agrupando-os aos já existentes.
- Foi inserida a opção *Importar Procedimentos*, a qual incorpora todos os procedimentos de um arquivo aos já existentes, caso exista no arquivo um procedimento com o mesmo nome de outro já presente, é pedida uma confirmação da substituição ao usuário.
- Ao abrir um arquivo apenas o cabeçalho dos procedimentos é analisado, como nos outros interpretadores, mas se um erro é encontrado o wxLogo ignora o procedimento e tenta encontrar outros que possam estar definidos posteriormente ao que contém erros, os demais ignoram o resto do arquivo.
- Quando é encontrado um erro de cabeçalho o wxLogo informa qual procedimento está errado e diz com detalhes qual o erro.

5.3.3 Linguagem

Foram feitas algumas adaptações na linguagem a fim de facilitar o entendimento e torná-la mais concisa. As formas originais também são suportadas.

- O comando de atribuição além da sintaxe original, aceita que a aspa no nome da variável seja fechada, ou que seja usada a notação mais comum para variáveis, usando o dois pontos no início do nome. Obtêm-se o mesmo resultado nas três formas: atribua "x 2, atribua "x"2 e atribua :x 2.

- O comando mudeposição, além de aceitar uma lista com dois números como parâmetro, permite que operações matemáticas sejam feitas dentro da lista ou ainda, que sejam passados dois valores diretamente ao invés de uma lista. As seguintes sintaxes resultam no mesmo movimento: mudeposição [156/2 17*3] e mudeposição 156/2 17*3.

- Os comandos que alteram as cores do lápis, do fundo e de preenchimento, aceitam um índice entre 0 e 15 que indicam cores pré-definidas ou uma lista contendo a intensidade de vermelho, verde e azul a ser usada. Se a lista possuir apenas um elemento ele é interpretado como um índice, se o índice for maior que 15 seu módulo na base 16 é usado, se dois valores forem passados na lista é assumido que o terceiro é zero.

- Foi inserido o comando *mudeel*, que muda a espessura do lápis, e os mnemônicos *mx* e *my* para os comandos *mudex* e *mudey*.

- O nome dos comandos, mnemônicos, estruturas e funções podem ser alterados no arquivo *wxLogo.ini*.

- Foram acrescentados os comparadores aritméticos $maiorouigual$ (>=) e $menorouigual$ (<=).

5.4 Recursos Disponíveis

Devido ao curto período de tempo disponível para a implementação do wxLogo, este ainda não oferece um leque de recursos tão vasto quanto outros interpretadores mais tradicionais. Apesar disso a fase de análise e interpretação foram desenvolvidas de maneira e permitir com facilidade a inclusão de novos comandos. Foram implementados principalmente recursos relativos à movimentação da tartaruga e aos recursos matemáticos.

5.4.1 Comandos Implementados

Os comandos implementados visam principalmente a movimentação da tartaruga e consequentemente a criação de desenhos na tela. Estes comandos são apresentados na tabela 5.1.

O comando pinte não está disponível no sistema *Linux* devido ao fato de sua implementação depender de funções que nas bibliotecas multiplataformas estão

disponíveis apenas para a plataforma *Windows*. São aguardadas as correções destas bibliotecas.

5.4.2 Funções Implementadas

Foi enfatizado no wxLogo a implementação de funções matemáticas. Estão presentes a maior parte das funções disponíveis em uma linguagem de programação. Foram implementados os operadores básicos: soma (+), subtração (-), multiplicação (*) e divisão (/), além das funções mostradas na tabela 5.2. Além dessas foram implementadas algumas funções que permitem uma maior interação do usuário com os procedimentos, como a caixasimnão [Título][Mensagem], que envia uma caixa de mensagem com os botões *sim* e *não*, e se o usuário clicar no *sim*, é retornado o valor *verd*, caso contrário *falso*, e a caixadequestão [Título][Mensagem], que envia uma caixa de mensagem com o espaço para que o usuário digite uma resposta, o texto digitado pelo usuário é retornado pela função.

5.4.3 Estruturas Implementadas

Foram implementadas as duas estruturas básicas mais utilizadas em uma linguagem de programação, e também na linguagem LOGO, são elas a estrutura condicional **se**, e estrutura de repetição **repita**, como mantêm a mesma sintaxe da linguagem original seu uso e exemplos podem ser vistos no capítulo anterior.

Comparadores Aritméticos

Os comparadores aritméticos podem ser usados para comparar dois números ou variáveis, o retorno é um valor booleano, geralmente usado por uma estrutura se. Foram implementados os comparadores mostrados na tabela 5.3.

Operadores Lógicos

A estrutura **se** suporta o uso de operadores lógicos, em LOGO estes operadores são usados na forma pré-fixa. No wxLogo foram implementados os seguintes operadores lógicos:

- e x y: Retorna $verdadeiro$ se ambos, x e y retornarem $verdadeiro$, caso contrário retorna $falso$. Ex.: se e 2=2 3=4 [pd 90][pe 90] (a segunda lista será executada).

- ou x y: Retorna $verdadeiro$ se x ou y retornarem $verdadeiro$, caso contrário retorna $falso$. Ex.: se ou 2=2 3=4 [pd 90][pe 90] (a primeira lista será executada).
- não x: Retorna $verdadeiro$ se x retornar $falso$ e vice-versa Ex.: se não 7>=5 [pd 90][pe 90] (a segunda lista será executada).

5.5 A Interface

A interface do wxLogo, mostrada na figura 5.1, é a mesma nos ambientes *Linux* e *Windows*, apesar disso apresenta algumas pequenas diferenças devido aos gerenciadores de janelas diferentes. A caixa de texto na parte inferior da interface é onde são digitados os comandos, para executá-los basta teclar *Enter* ou clicar no botão *Executar*.

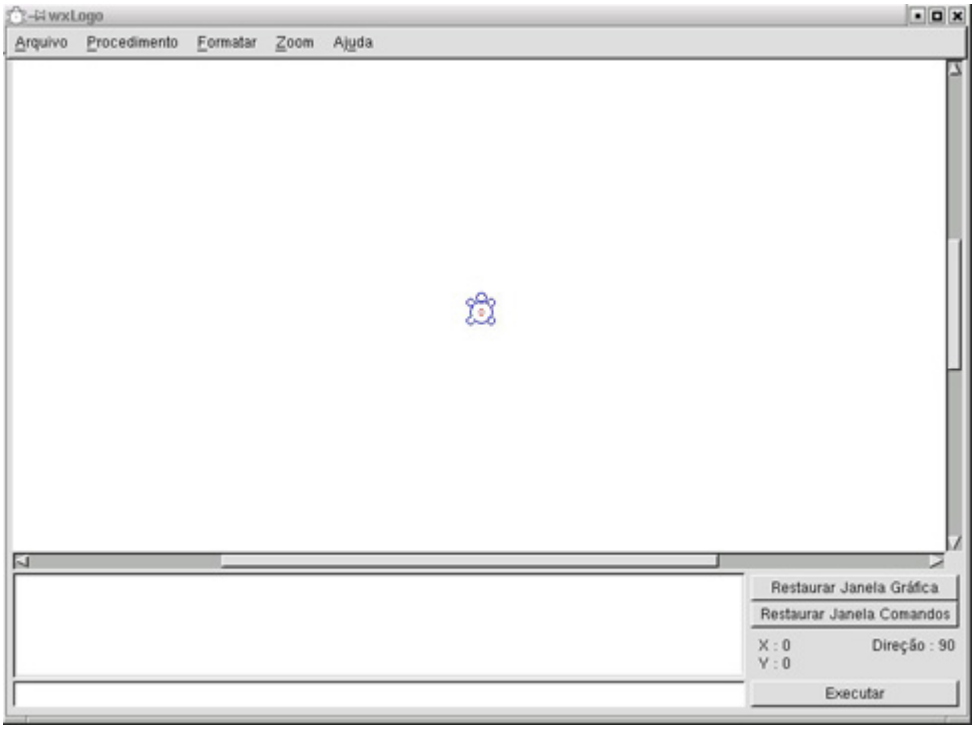

Figura 5.1: Interface do wxLogo no sistema *Linux*

5.5.1 Procedimentos do Usuário

Um dos recursos mais importantes da linguagem LOGO é a inclusão de novos comandos pelo usuário, esses comandos são chamados de *procedimentos*. Foi implementado no wxLogo o suporte à recursão, que pode ser feita através de procedimentos que executam a si próprios, é importante lembrar também que as variáveis criadas dentro de um procedimento são locais, assim são retiradas da memória quando o procedimento termina, ao contrário das variáveis criadas através da caixa de entradas, que são globais e ficam na memória até o wxLogo ser fechado.

Criando Novos Procedimentos

Para criar um procedimento existem duas alternativas, ou usar o comando *aprenda*, ou usar o *menu Procedimento* onde existe a opção *Novo*. Em qualquer dessas opções o editor de procedimentos será aberto, já com a estrutura do procedimento pronta, como mostra a figura 5.2. É permitido alterar o nome e usar quantos parâmetros forem necessários. Os comandos devem ser digitados entre as instruções *aprenda* e *fim*, qualquer instrução digitada fora do limite dessas instruções será ignorado. Ao terminar de digitar o código é preciso atualizar a área de trabalho para que o novo procedimento seja analisado e reconhecido. Ao inserir um novo procedimento, apenas seu cabeçalho é analisado, caso este contenha algum erro todo o procedimento é ignorado, caso contrário ele é inserido e na decorrência de sua primeira execução seu código será analisado.

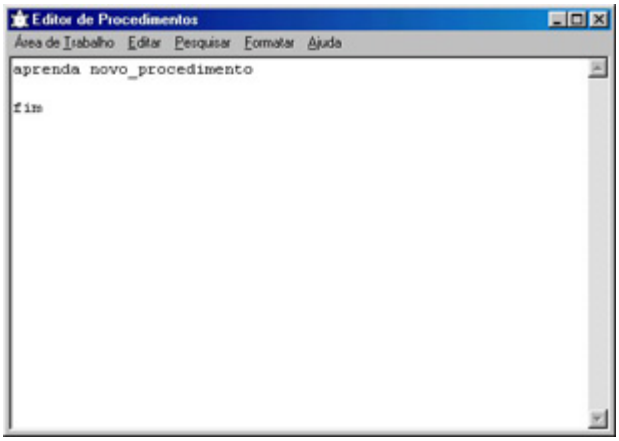

Figura 5.2: Editor de Procedimentos

Editando Procedimentos

Para editar um procedimento existem também duas opções, ou usar o comando edite ou a opção *Editar...* no *menu Procedimento*. Caso seja usado o menu ou o nome do procedimento passado ao *edite* seja inválido, será aberta uma lista contendo o nome dos procedimentos existentes como mostrado na figura 5.3. Existe também na lista a opção para editar todos, opção que também está presente no *menu Procedimento*.

Figura 5.3: Editar Procedimentos

Removendo Procedimentos

Para remover procedimentos já existentes existe a opção *Remover...* no *menu Procedimento*. Esta opção abre uma lista com todos os procedimentos, semelhante à da figura 5.3, para que um deles seja selecionado e removido. Existe também a opção de remover todos, tanto no *menu* como na lista.

5.5.2 Salvando Arquivos

Para que os procedimentos possam ser usados posteriormente, eles devem ser salvos em um arquivo. No *menu Arquivo* existe a opção *Salvar*, a qual salva todos os procedimentos existentes em um arquivo com a extensão *lgo*. Esta opção não salva o ambiente, se for necessário salvar o desenho feito pela tartaruga a opção *Salvar Bitmap* no *menu Arquivo* deve ser usada. Ela salva toda a área desenhada em um arquivo.

5.5.3 Abrindo Arquivos

Quando o wxLogo é iniciado a tartaruga conhece apenas os comandos básicos, a opção *Abrir* no *menu Arquivo*, permite que seja aberto um arquivo contendo procedimentos salvos em outra sessão. Caso já existam procedimentos definidos, estes serão removidos antes que os novos sejam carregados.

5.5.4 Importando Procedimentos

Para que os procedimentos contidos em um arquivo sejam agrupados com os já definidos até o momento a opção *Importar Procedimentos* pode ser usada. Caso já exista um comando com o mesmo nome de outro contido no arquivo, é enviada uma caixa de mensagem que permite o usuário substituir ou não este procedimento.

5.6 A Ajuda

Os arquivos de ajuda do wxLogo foram feitos no formato *html*, isso permite que mesmo sem executar o programa, os arquivos de ajuda sejam visualizados em qualquer navegador de Internet. Além disso isso possibilita que estes arquivos sejam facilmente disponibilizados em uma *homepage*. O wxLogo possui seu próprio navegador, como pode ser visto na figura 5.4. Para abrir o índice da ajuda, use a tecla de atalho **F1** ou o *menu Ajuda*. Neste *menu* existem ainda as opções *Lista de Comandos* que é um atalho para o item mais usado, e *Sobre*, que contém informações sobre o projeto wxLogo.

A ajuda do wxLogo é divida em 7 tópicos:

- **O que é LOGO**: Neste tópico são apresentados a história da linguagem e seus fundamentos teóricos.

- **Conhecendo a Interface**: São descritos os itens da interface, suas funções e recursos, e também as opções do $menu$.

- **Comandos Básicos**: É fornecida uma lista com os comandos básicos da linguagem, que são também os mais utilizados, visando principalmente os usuários iniciantes. São apresentados a descrição, a sintaxe e os exemplos de cada comando.

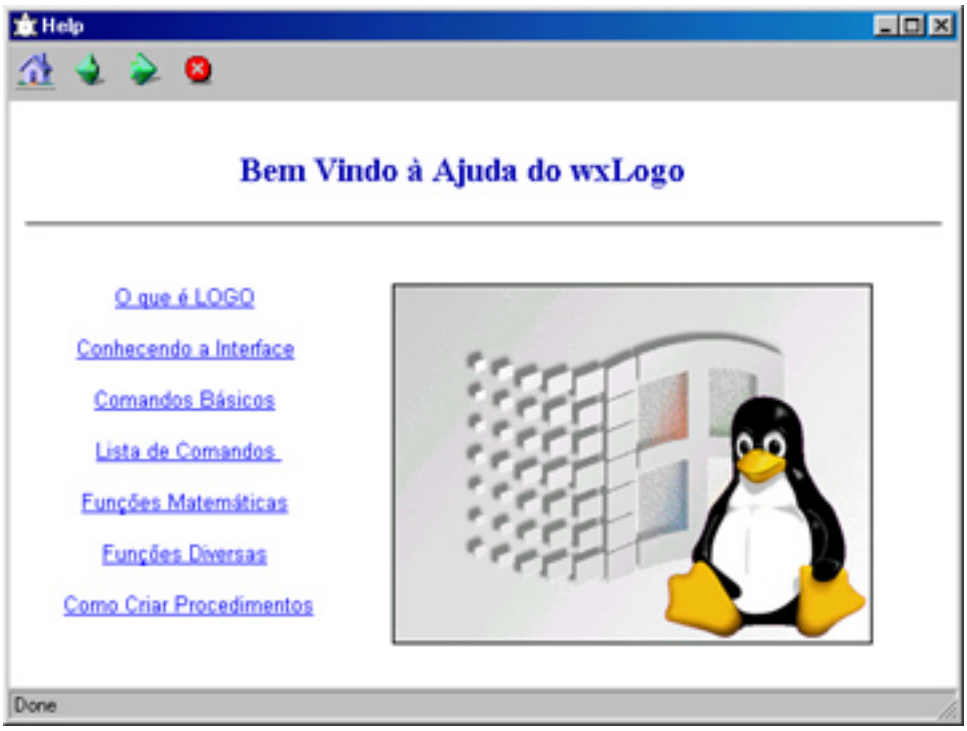

Figura 5.4: Ajuda do wxLogo

- **Lista de Comandos**: É fornecida uma lista completa de todos os comandos oferecidos pelo wxLogo com links que levam à descrição, sintaxe e exemplo de cada comando.
- **Funções Matemáticas**: É fornecida uma lista com as funções matemáticas oferecidas, além da descrição são mostrados exemplos.
- **Funções Diversas**: É fornecida uma lista de outras funções além das matemáticas, suportadas pelo wxLogo, com sintaxe e exemplos.
- **Como Criar Procedimentos**: Um guia de como criar, editar, remover, salvar e abrir procedimentos.

Comando	*Mnemônico*
parafrente	pf
paratrás	pt
paradireita	pd
paraesquerda	pe
paracentro	pc
espere	
pare	
uselápis	ul
usenada	un
useborracha	ub
mudex	mx
mudey	my
mudeposição	mudepos
arco	
circunferência	circ
elipse	
rotule	
pinte*	
mudecl	
mudeel	
mudecf	
mudecp	
escreva	esc
mostre	
atribua	atr
aprenda	apr
edite	ed

Tabela 5.1: Comandos Implementados no wxLogo

Função	*Exemplo*	*Resultado*
sen	sen 30	0.5
cos	cos 30	0.866025
tan	tan 30	0.57735
arcsen	arcsen 0.5	30
arccos	arccos 0.866025	30
arctan	arctan 0.57735	30
pi	pi	3.14159265358979
potência	potência 2 3	8
raizq	raizq 64	8
ln	ln 100	4.60517
log10	log10 100	2
exponencial	exponencial 1	2.71828
oposto	oposto 2	-2
resto	resto 10 3	1
abs	abs -10	10
inteiro	inteiro 10.7	10
arredonde	arredonde 10.7	11
sorteienúmero	sorteienúmero 100	Um número entre 0 e 100
sortnum	sortnum 100	Um número entre 0 e 100

Tabela 5.2: Funções Matemáticas Disponíveis no wxLogo

Comparador	*Exemplo*	*Resultado*
=	mostre 2=1+1	*verd*
>	mostre 4/2>1+1	*falso*
<	mostre resto 10 7 < 4	*verd*
>=	mostre 2*2>=4+2	*falso*
<=	mostre 2*2<=4+2	*verd*

Tabela 5.3: Operadores Aritméticos Implementados no wxLogo

Capítulo 6

Comentários Sobre a Implementação

6.1 Introdução

O wxLogo foi desenvolvido a partir de recursos disponíveis gratuitamente e com suporte a várias plataformas. Durante seu desenvolvimento foram aplicados conceitos de *linguagens formais e autômatos, algoritmos e estruturas de dados, técnicas de programação* e *compiladores* entre outros. Neste capítulo serão apresentadas uma breve descrição da implementação da interface, as estruturas e formalismos utilizados e os problemas encontrados durante a implementação.

6.2 Recursos Utilizados

O wxLogo foi implementado usando-se a linguagem de programação C++ e o paradigma de orientação a objetos. O compilador utilizado foi o *gcc* (*Gnu C Compiler*) o qual possui versões para *Linux* e *Windows*, disponível gratuitamente em `http://www.gnu.org`. Para a implementação da parte gráfica, foram usadas as bibliotecas *wxWindows*, disponíveis gratuitamente em `http://www.wxwindows.org`. Estas bibliotecas suportam desenvolvimento multi-plataforma, e foram escolhidas com base nos estudos de [Gomes (2000)]. As classes foram criadas de maneira a deixar as funções de análise e interpretação, independentes da interface gráfica.

6.3 Interface

6.3.1 Descrição Geral

A interface gráfica foi criado sobre um *frame* dividido em três partes, uma janela com barras de rolagem, onde são feitos os desenhos e que ocupa dois terços da parte superior da tela, e dois painéis na parte inferior, um ao lado do outro. Sobre o primeiro painel são criadas as duas caixas de texto uma sobre a outra, e sobre o segundo três botões e três caixas de texto estáticas.

6.3.2 Atualização da Tela

Sempre que a tela de desenho é movimentada ou outra janela é colocada em sua frente, ela deve ser redesenhada, para que não seja necessário refazer todo o desenho, reinterpretando todos os movimentos da tartaruga, é guardado na memória um desenho idêntico ao da tela, sempre que for necessário atualizar a tela, as coordenadas da área que está sendo visualizada são calculadas e esta área é copiada do desenho em memória. Quando a tartaruga se movimenta também é preciso atualizar tela, para que a posição onde estava seja redesenhada. Para isso primeiro se desenha o que o comando pede, depois calcula-se a área onde a tartaruga estava e copia-se apenas esta área da imagem em memória, depois a tartaruga é desenhada na nova posição.

6.3.3 Observações Importantes

Para modificar o código ou interface do wxLogo devem ser tomados alguns cuidados, e observados alguns detalhes importantes. Entre eles destacam-se:

É permitido ao usuário redimensionar a janela principal, quando isto é feito é preciso recalcular o tamanho dos outros objetos da interface. É preciso controlar a primeira execução do evento de redimensionamento, por que este é chamado pelo *Windows* quando a janela principal é criada, como os outros componentes ainda não foram criados esta execução resulta em um erro fatal. Para este controle foi usada uma variável global booleana, iniciada como *falso*, e que assume *verdadeiro* depois da primeira execução.

As coordenadas da tartaruga devem ser adaptadas antes de irem para a janela de desenho, a posição $0, 0$ desta janela é seu canto superior esquerdo, enquanto a da tartaruga deve ser o meio desta mesma janela. Além disso o eixo Y cresce na direção descendente, ao contrário do que ocorre normalmente. Para resolver este problema são usados fatores de correção, que são calculados em função do

tamanho da janela. As coordenadas são armazenadas normalmente, e no momento de irem para a tela são aplicados os fatores de correção a cada eixo da coordenada.

É importante lembrar de usar a função *Skip()* sobre um evento que é interceptado, evitando assim que este evento seja repassado para as outras janelas e cause um comportamento indesejável da interface.

6.4 Fase de Análise

A fase de análise foi dividida em análise léxica e sintática, não houve implementação de análise semântica porque seria muito complexa e de pouca utilidade, já que em LOGO não existem declarações ou tipos de variáveis e não existe uma ordem formal para a execução dos procedimentos, o usuário pode chamar qualquer procedimento a qualquer momento. É verificado apenas se as variáveis utilizadas foram criadas, local ou globalmente.

6.4.1 Análise Léxica

Na análise léxica o código recebido é separado em *símbolos*. Cada *símbolo* é classificado de acordo com os tipos da tabela 6.1, depois é armazenado em uma tabela. Cada Procedimento possui uma tabela léxica própria, a qual é criada na primeira vez que o procedimento é executado, caso o procedimento seja alterado, sua tabela será apagada, ela será reconstruída novamente da próxima vez que ele for executado. Os comandos digitados na caixa de entrada são classificados em uma tabela léxica temporária.

6.4.2 Análise Sintática

A análise sintática recebe a *tabela de símbolos* construída na análise léxica, a análise é feita através da seguinte gramática:

G = (Variáveis, Terminais, Produções, SímboloInicial) onde:

SímboloInicial = {LISTA_CMDS}

Variáveis = {LISTA_CMDS, CMD, PARS, EXPRESSAO, FUNCAO, SE, CONDICAO, LISTA_OPC, REPITA, ATRIB, APRENDA, LISTA_VARS, CONSULTA, LISTA_STRING, LISTA_VAL}

Tipo	*Descrição*
command	Comando nativo do LOGO
function	Função nativa do LOGO
procedure	Procedimento do usuário
loop	Estrutura de repetição (repita)
if	Estrutura condicional (se)
attribute	Comando de atribuição (atribua)
learn	Comando aprenda
query	Comando de consulta (mostre, escreva)
stringlist	Comando que recebe duas listas de strings (caixasimnão)
vallist	Comando que recebe uma lista de valores (mudepos, mudecl, etc.)
operator	Operador matemático ($+, -, *, /$)
comparator	Comparador ($=, <, >, <=, >=$)
logical	Operador lógico ($e,\ ou$)
not	Negação lógica (não)
variable	Variável
value	Valor numérico
string	Tipo $string$
boolean	Tipo booleano (verdadeiro ou falso)
openlist	Início de lista "["
closelist	Fim de lista "]"
openpar	Início de bloco entre parênteses "("
closepar	Fim de bloco entre parênteses ")"
invalid	Tipo não identificado

Tabela 6.1: Tipos da Tabela Léxica

Terminais = {command, function, procedure, loop, if, attribute, learn, query, stringlist, vallist, operator, comparator, logical, not, variable, value, string, boolean, openlist, closelist, openpar, closepar, invalid}

Produções = {
LISTA_CMDS $\Rightarrow$ CMD LISTA_CMDS | ε,
CMD $\Rightarrow$ command PARS | procedure PARS | SE | REPITA | ATRIB | APRENDA | CONSULTA | LISTA_STRING | LISTA_VAL | ε ,
PARS $\Rightarrow$ EXPRESSAO PARS | ε ,
EXPRESSAO $\Rightarrow$ EXPRESSAO operator EXPRESSAO | openpar EXPRESSAO

closepar | FUNCAO | variable | value | string | boolean,
FUNCAO $\Rightarrow$ function PARS,
SE $\Rightarrow$ if CONDICAO openlist LISTA_CMDS closelist LISTA_OPC,
CONDICAO $\Rightarrow$ EXPRESSAO comparator EXPRESSAO | openpar CONDICAO closepar | logical CONDICAO CONDICAO | not CONDICAO,
LISTA_OPC $\Rightarrow$ openlist LISTA_CMDS closelist | ε,
REPITA $\Rightarrow$ loop EXPRESSAO openlist LISTA_CMDS closelist,
ATRIB $\Rightarrow$ attribute string EXPRESSAO | attribute variable EXPRESSAO,
APRENDA $\Rightarrow$ learn invalid LISTAVARS | learn procedure LISTAVARS,
LISTA_VARS $\Rightarrow$ variable LISTA_VARS | ε,
CONSULTA $\Rightarrow$ query EXPRESSAO | query CONDICAO,
LISTA_STRING $\Rightarrow$ stringlist openlist string closelist openlist string closelist,
LISTA_VAL $\Rightarrow$ vallist openlist PARS closelist | vallist PARS,
}

Quando um erro é encontrado a execução de todo o código é parada, inclusive a análise, e uma mensagem bastante específica sobre o erro é enviada ao usuário. Como em LOGO não existem separadores de comandos e parâmetros, ou marcadores de fim de linha, é inviável tentar analisar o código adiante, isso geraria uma quantidade muito grande de mensagens que deixaria o usuário confuso.

6.5 Fase de Interpretação

A fase de interpretação recebe a tabela de símbolos já analisada, ou seja, não há preocupação com erros no código. Esta fase é dividida em duas partes distintas; uma executa as estruturas (condicional e repetição), substitui as variáveis por seu valor correspondente, resolve as expressões matemáticas e funções; outra recebe o nome do comando e o valor dos parâmetros, e executa este comando, ao final, caso necessário, avisa à interface para que esta se atualize. Todo comando que cria algum tipo de desenho na interface, ou movimentam a tartaruga, têm seus dados armazenados em um vetor, assim caso necessário pode-se redesenhar toda a tela, como por exemplo quando se muda a cor do fundo. Os atributos guardados são:

- Tipo do comando (linha, texto, circunferência, arco, elipse ou preenchimento).
- Posição e direção da tartaruga.
- Cor, espessura e tipo do lápis (sem lápis, com lápis ou com borracha).

- Raio (para circunferência)
- Raio e ângulo (para arco)
- Raio maior e raio menor (para elipse).
- Texto (para o comando rotule).

6.6 Arquivo de Configuração

Uma novidade no wxLogo é a possibilidade de configurar o nome dos comandos, permitindo assim que sejam feitas alterações a fim de abrir arquivos de outros interpretadores que possuam pequenas diferenças no nome dos comandos. Além disso facilita a possível criação de versões em outras línguas. No arquivo *wxLogo.ini* encontram-se as definições dos nomes dos comandos, caso o nome de algum comando não seja encontrado e esteja com a sintaxe errada o valor padrão é usado. A sintaxe do arquivo de configuração é a seguinte: *#COMANDO="nome"*, onde $COMANDO$ é o nome padrão do comando escrito em maiúsculas e sem assentos, e *"nome"* é o nome atribuído a este comando, por exemplo:
#SE="se"
#PARATRAS="paratrás".

6.7 Problemas na Implementação

O principal problema encontrado na implementação decorreu da indisponibilidade e de funções nas bibliotecas *wxWindows* para o ambiente *Linux*. O comando *pinte* está funcionando apenas para o ambiente *Windows*, já que para *Linux* não são fornecidas funções para ler um determinado ponto na tela, sendo assim, não há como implementá-lo. A ausência desta função já foi reclamada aos desenvolvedores das bibliotecas, assim que forem disponibilizadas em novas versões o problema será sanado.

Existem também funções e objetos que apresentam comportamento e aparência diferentes nos dois sistemas, o que obrigou o uso de diretivas de compilação em algumas partes do código relacionadas à formatação da interface.

Capítulo 7

Conclusão

No desenvolvimento desta monografia foram realizados estudos sobre o construtivismo, o construcionismo, o uso dos computadores na educação e a linguagem de programação LOGO. Através destes, foi possível verificar que existe uma carência de *software* desenvolvidos com a proposta de mudar e melhorar o sistema de ensino como existe hoje. A maior parte dos *software* educacionais apenas automatizam o sistema já existente, e não possuem um embasamento teórico pedagógico. A linguagem LOGO além de possuir um forte embasamento teórico, propõe uma nova filosofia no ensino, com mudanças na relação aluno-professor, na reação mediante um erro, e na maneira como os alunos são divididos em turmas. Foi observado também que a maior parte destes *software*, inclusive os interpretadores LOGO, são desenvolvidos apenas para o ambiente *Windows*, o que não possibilita aos usuários ou instituições que os adotem, a liberdade de avaliar e escolher seu sistema operacional, obrigando-os a adquirir um sistema caro, principalmente se levar-se em conta a situação financeira das instituições públicas de ensino.

Estes estudos motivaram o desenvolvimento do wxLogo, um interpretador multi-plataforma para a linguagem LOGO, apresentado no capítulo 5. Durante seu desenvolvimento foi possível observar que é possível desenvolver um *software* multi-plataforma adotando-se apenas recursos disponíveis gratuitamente. Também foi constatado que as bibliotecas *wxWindows*, possuem um excelente manual de referência e uma vasta e poderosa coleção de recursos, mas possuem algumas funções que são disponíveis apenas para um determinado sistema, o que limita um pouco a portabilidade do código. Além disso a versão para o sistema *Windows* mostrou-se mais rápida para a manipulação da interface que a versão do sistema *Linux*.

Sugestões para trabalhos futuros:

- Disponibilizar o wxLogo em outros idiomas.
- Fazer a caixa de entrada e o editor de procedimentos aplicarem cores que destaquem os comandos, funções e estruturas.
- Fazer com que as configurações feitas pelo usuário sejam gravadas no arquivo de configuração, e restauradas na próxima execução.
- Permitir que o arquivo de configuração contendo o nome dos comandos seja configurado pelo usuário.
- Implementar um editor gráfico para o arquivo de configuração.
- Implementar as funções de tratamento de lista.
- Implementar recursos de multimídia.
- Implementar a estrutura *enquanto*.
- Implementar comandos em 3D.
- Implementar comandos que controlem as saídas serial e paralela do computador.
- Criar um assistente inteligente que auxilie o usuário a corrigir seus erros.
- Criar uma tartaruga robô que possa atender aos comandos básicos de movimento da linguagem LOGO, e seja controlada por ondas de rádio.
- Oferecer suporte ao paradigma de orientação a objetos.
- Compilar e testar o wxLogo em mais sistemas operacionais.
- Fazer um estudo sobre a situação dos *software* educacionais e sistemas operacionais usados nas escolas de Lavras ou de toda a região.
- Fazer parceria com escolas da comunidade para ensinar a linguagem LOGO em algumas turmas, e acompanhar seu rendimento.

Referências Bibliográficas

[Aho & Sethi & Ullman (1995)] Aho, Alfred V. & Sethi, Ravi & Ullman, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas* 1.ed. Rio de Janeiro, LTC, 1995.

[Campos (1997)] Campos, Márcia de Borba *Construtivismo* Porto Alegre, CPGIE-UFRGS, 1997, url: `http://www.penta.ufrgs.br/~marcia/construu1.htm`

[Chaves (1999)] Chaves, Eduardo O. C. *O Computador na Educação*. 1999, url: `http://www.edutecnet.com.br/Textos/Self/EDTECH/funteve.htm`

[Correia et alii (1999)] Correia, L. H. & Amaral, K. C. A. & Uchôa, J. Q. & Costa, H. A. X. *Computador tutelado*. Lavras, UFLA/FAEPE, 1999.

[Deitel & Deitel (2001)] Deitel, Harvey. M. & Deitel, Paul. J. *C++ Como Programar* 3.ed. Porto Alegre, Bookman, 2001.

[Empório Linux (2002)] Empório Linux *Sobre o Linux* 2002, url: `http://www.emporio-linux.com.br/sobrelinux/`

[FAC (2002)] Fundaustral Asociación Civil *Algunos Motivos Por los Cuales Incorporar el Trabajo con LOGO en la Escuela* 2002, url: `http://www.nalejandria.com/fundaustral/trabaj/porque.htm`

[Flores (1996)] Flores, Angelita Marçal *A Informática na Educação: Uma Perspectiva Pedagógica*. Monografia para obtenção do título de Especialista em Informática. Tubarão, Universidade do Sul de Santa Catarina, 1996.

[Franco (2001)] Franco, Sérgio Roberto Kieling *Contribuições de Piaget e Vygotsky Para a Educação* Porto Alegre, UFRGS, 2001, url: http://www.pgie.ufrgs.br/~franco/textos/piavygo.htm

[GNU (2001)] GNU's Not Unix! *GNU General Public License - GNU Project - Free Software Foundation (FSF)* 2001, url: http://www.gnu.org/copyleft/gpl.html

[Gomes (2000)] Gomes, Renato de Souza *Pesquisa de Bibliotecas Multiplataforma para Programas Multimídia* Monografia para obtenção do título de Bacharel em Ciência da Computação, Lavras, DCC-UFLA, 2000.

[Menezes (2001)] Menezes, Paulo Blauth *Linguagens Formais e Autômatos* 4.ed. Porto Alegre, Sagra Luzzatto, 2001.

[Nied (2001)] Nied (2001) *Núcleo de Informática Aplicada à Educação*, UNICAMP, Campinas, url: http://www.nied.unicamp.br/

[Papert (1980)] Papert, Seymour *Mindstorms: children, computers and powerful ideas*. New York, Basic Books, 1980.

[Papert (1986)] Papert, Seymour *Constructionism: A New Opportunity for Elementary Science Education. A proposal to the National Science Foundation*. Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group, Cambridge, Massachusetts, 1986.

[Papert (1994)] Papert, Seymour *A máquina das crianças: repensando a escola na era da informática*. Porto Alegre, Artes Médicas, 1994.

[Papert (1999)] Papert, Seymour *Diversity in Learning: A Vision for the New Millennium* 1999, url: http://www.papert.com/articles/diversity/DiversityinLearningPart1.html

[Polônia (2000)] Polônia, Eunice *Teorias Interacionistas - Principais Diferenças entre Piaget e Vygotsky* url: http://www.pgie.ufrgs.br/listas/pca-l/doc00003.doc

[Ramos (1996)] Ramos, Edla Maria Faust *Análise Ergonômica do Sistema Hiper-Net Buscando o Aprendizado da Cooperação e da Autonomia* Florianópolis, UFSC, cap.5, 1996. (Tese de Doutorado)

[Santos (2000)] Santos, Nilson Moutinhos dos *I.A. Voltada à Educação: Linguagem Logo* Grupo de Sistemas Inteligentes, 2000, url: `http://www.din.uem.br/ia/a_correl/iaedu/`

[Silva (2001)] Silva, Maria Beatriz Costa Cabral Costa *Melhorias das Condições de Ensino: Recomendações para o Funcionamento dos Ambientes Informatizados* Porto Alegre, UFRGS, 2001 (Monografia para obtenção do título de Especialista em Informática na Educação)

[Smart et alii (2001)] Smart, Julian et alii *wxWindows 2.2: A portable C++ and Python GUI toolkit* Artificial Intelligence Applications Institute, 2001, url: `http://www.wxwindows.org`

[Souza (1998)] Souza, Rosemeire Silva e *A Escolha do Software Educativo*. 1998, url: `http://www.geocities.com/Athens/Ithaca/8750/index.html`

[Time (1999)] Papert, Seymour Papert on Piaget *Time Magazine's Special Issue: The Century's Greatest Minds* March 29, p.105, 1999.

[Valente (1996)] Valente, José Armando et alii. *O professor no ambiente Logo: Formação e atuação*. Campinas, NIED-Unicamp, 1996.

[Valente (2001)] Valente, José Armando *Informática na Educação: Instrucionismo x Construcionismo* 2001, url: `http://www.divertire.com.br/artigos/artigos.htm`

[Zacharias (1998)] Zacharias, Vera *A Linguagem Logo* Centro de Referência Educacional, 1998, url: `http://members.tripod.com/lfcamara/linlogo.html`