Universidade Federal de Uberlândia

Faculdade de Computação

Programa de Pós-Graduação em Ciência da Computação

Realização de trabalhos de laboratório online e utilização de materiais auto-normalizantes no Ensino à Distância

Junia Magalhães Rocha

Fevereiro, 2009

Dados Internacionais de Catalogação na Publicação (CIP)

R672r Rocha, Junia Magalhães, 1985-
Realização de trabalhos de laboratório online e utilização de materiais auto-normalizantes no ensino à distância / Junia Magalhães Rocha. - 2009.
78 f. : il.

Orientador: Antônio Eduardo Costa Pereira.
Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Linguagem de programação (Computação) - Teses. 2. Ensino a distância - Teses. I. Pereira, Antônio Eduardo Costa, 1948- II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3.06:800.92

Elaborada pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

Universidade Federal de Uberlândia

Faculdade de Computação

Programa de Pós-Graduação em Ciência da Computação

Realização de trabalhos de laboratório online e utilização de materiais auto-normalizantes no Ensino à Distância

Dissertação apresentada por Junia Magalhães Rocha à Universidade Federal de Uberlândia (UFU) como parte dos requisitos para obtenção do título de Mestre.

Banca Examinadora:

Antônio Eduardo Costa Pereira, Dr. Orientador

Guilherme Bittencourt, Dr. UFSC

Luciano Vieira Lima, Dr. UFU

Realização de trabalhos de laboratório online e utilização de materiais auto-normalizantes no Ensino à Distância

Junia Magalhães Rocha

Dissertação apresentada por Junia Magalhães Rocha à Universidade Federal de Uberlândia (UFU) como parte dos requisitos para obtenção do título de Mestre.

Antônio Eduardo Costa Pereira, Dr.	Prof. Sandra Aparecida de Amo, Dra.
Orientador	Coordenadora do
	Curso de Pós-Graduação

Dedico esse trabalho aos meus pais, José Antônio e Maria das Graças, a Vovó Nini, a minha irmã Juliane e ao Fábio.

Agradecimentos

Em primeiro lugar agradeço a Deus pela força que me concedeu para a realização do mestrado. Aos meus pais, José Antônio e Dadá por investirem na concretização desse sonho e caminhar comigo lado a lado. A minha querida afilhada e irmã Juliane, que muito me ajudou nessa jornada no mestrado. A minha querida Vovó Nini pelas inúmeras orações. Vocês sorriram e choraram comigo. Sou eternamente grata!

Agradeço à minha família pela compreensão nos momentos que estive ausente. Ao meu primo Thiago Magalhães que me incentivou nos momentos difíceis. E a todos que direto ou indiretamente me acompanharam nesses dois anos. Ao Fábio Carneiro que sempre me apoiou e me incentivou nos estudos. Obrigada pelo carinho e pela companhia.

Aos meus amigos e principalmente a Leni Coelho pelo companheirismo e amizade nas horas que mais precisei.

Ao Prof. Dr. Eduardo Costa, pelo gigantesco conhecimento. Esses dois anos sob sua orientação me proporcionou uma oportunidade ímpar. Ao Prof. Dr. Luciano Vieira, pelos valiosos ensinamentos, consolos e por me acolher sempre. Seremos sempre amigos!

Ao Prof. Dr. Guilherme Bittencourt, por aceitar o convite para participar da banca e pelas contribuições.

Aos meus amigos do LABIA que tanto me apoiaram, presentes nos momentos de café, almoço e muito trabalho. Reny Cury, Reane Gourlart, Hipólito Barbosa, João Barbosa, Matheus Garcia, Daniel Cardoso, Breno Finotti, Everton Dias, Fernando, George, Simonini e Luciano Vieira, obrigada por tudo!

Aos professores, Profa. Rita Julia, Prof. Marcelo Maia, Prof. Pedro Frosi, Prof. João Nunes, Prof. Jamil e Prof. Camacho que me passaram bastante conhecimento no decorrer das disciplinas. Aos meus colegas de mestrado e ao apoio financeiro da FAPEMIG.

Sumário

Lista de Figuras

Resumo

Essa pesquisa tem como objetivo propor formas alternativas de realização de cursos de graduação à distância, promovendo uma qualidade do mesmo. Essa temática é relevante para o meio acadêmico em razão de se discutir melhorias na forma de ensino baseados no uso de materiais auto-normalizantes e a prática de laboratórios online. Esses adendos propostos visam assegurar oportunidades aos alunos, ampliando as fronteiras e contribuindo na democratizando do ensino. Espera-se que com essas propostas seja possível melhorar a qualidade do ensino à distância, permitindo um desenvolvimento do aluno com vivências práticas e que, com o material utilizado, seja possível avançar no conhecimento com redução das possibilidades de falhas.

Palavras-Chave

Ensino à Distância, Laboratórios Online e Materiais Auto-normalizantes

Abstract

This study has, as its objective, the proposal of alternative ways of taking Distance Graduation Courses, emphasizing their quality. This theme is relevant in the academic circle because it discusses improvements in teaching methods based on the use of Self Normalizing Materials and the practicability of online laboratories. These addendum have as an objective the guaranteeing of student opportunities, opening up frontiers and contributing to the democratization of teaching. It is hoped that these proposals will make it possible to improve the quality of distance learning, permitting student development with practical experiences and, with the material utilized, making it possible to increase in knowledge while reducing the possibility of failures.

Keyword

Distance Learning, Online Laboratories, Self Normalizing Materials.

Capítulo 1

Introdução

Atualmente, no Brasil, as bases legais para a modalidade de Educação à Distância (EAD) foram estabelecidas pela Lei de Diretrizes e Bases da Educação Nacional (Lei n.º 9.394, de 20 de dezembro de 1996), regulamentada pelo Decreto n.º 5.622, publicado no D.O.U. de 20/12/05. Esta Lei informa que os cursos de graduação e educação tecnológica para serem feitos à distância precisam de um credenciamento específico junto à União. Os cursos de Direito, Odontologia, Medicina e Psicologia necessitam ainda de uma manifestação dos seus respectivos conselhos de classe, ou seja, uma aprovação. Na atualidade, 20% da carga horária de cursos de graduação podem ser realizados à distância, segundo regulamentação do MEC (Portaria MEC nº. 4.059/04).

A modalidade de Educação à Distância vem conquistando espaço no Brasil, se consolidando e vencendo resistências. Dados do Censo da Educação Superior de 2006 revelam que de 2003 a 2006, os cursos de graduação cresceram 571%. Essa expansão acelerada se deve em grande parte a políticas públicas de ensino.

Apesar de todo esse crescimento, essa modalidade de educação ainda se depara com certas dificuldades. Algumas críticas tais como: falta de contato físico dos estudantes com a sociedade acadêmica e os colegas, ausência de um professor presencial para verificar o desempenho dos alunos, ausência de trabalhos práticos dentre outros, são temas constantemente citados por especialistas e pela sociedade como pontos fracos do EAD.

Hoje, o EAD tem conseguido atender às necessidades dos profissionais que não tem disponibilidade para estar em um curso presencial. Assim é preciso propor possibilidades

para que esses estudantes possam adquirir uma formação sólida. Desta forma, este trabalho de mestrado propõe enfoques diferentes dos comumente utilizados pela academia brasileira. Uma possibilidade é a utilização de materiais auto-normalizantes que possibilitarão ao aluno a oportunidade de descobrir falhas durante o seu aprendizado corrigindo-as em tempo, evitando, desta forma, que conhecimentos errôneos se consolidem.

A segunda possibilidade de ensino que será discutida apóia a aprendizagem pelo uso de laboratórios à distância online. Os experimentos construídos pelos alunos podem estar em uma Universidade (distante dele) ou próximos do aluno. Esse trabalho incentiva a realização de experimentos próximos ao aluno, observados e corrigidos por um professor à distância. Com essa abordagem o aluno poderá demonstrar, na prática, os conceitos e princípios vistos na teoria.

Assim, os objetivos desse trabalho são: 1. apresentar uma proposta de material que pode ser aplicado em cursos à distância baseado no conceito de auto-normalizante; 2. apresentar exemplos de aulas práticas em cursos à distância; 3. discutir a importância de atividades práticas por meio de laboratórios online em cursos à distância; 4. apresentar uma introdução da linguagem de programação Bigloo, fornecendo ao leitor conhecimentos básicos para implementação dos exemplos descritos nos capítulos 3 e 4.

A finalidade ao se desenvolver esse trabalho é mostrar a necessidade de aperfeiçoamento do EAD, o qual vem crescendo aceleradamente no Brasil e no exterior. Através de propostas que visam minimizar os principais problemas enunciados pela sociedade que o descredencia. Assim será possível uma maior oferta de cursos à comunidade sem perda na qualidade de ensino.

Para o desenvolvimento deste trabalho será utilizada a linguagem de programação Bigloo. Ela é uma implementação do Scheme, um dialeto específico do Lisp. O ambiente do Bigloo permite um constante diálogo entre o programador e o computador. Com uma notação simples e poderosa, é possível construir grandes aplicações e com isso criar programas que geram outros programas. Serão feitos alguns exemplos em Bigloo para verificar como aplicar os conceitos de laboratórios à distância online e materiais auto-normalizantes em cursos de graduação.

Esse trabalho será dividido nos seguintes capítulos. No capítulo 2 será discutida a linguagem de programação reflexiva, Scheme, utilizando o dialeto Bigloo; com a instalação

do compilador da linguagem até o desenvolvimento de alguns programas-exemplos para estudo da mesma. Já no capítulo 3 será apresentada uma aplicação da teoria de materiais auto-normalizantes, permitindo uma melhor compreensão do que são, por que são importantes e as formas de detecção e correção de erros no conhecimento dos alunos. No capítulo 4 será mostrada a participação de estudantes em atividades de laboratório online e observações científicas. Esse capítulo mostra a importância da realização de atividades práticas e exemplos. A teoria apresentada nas aplicações exemplo, tais como, Métodos Numéricos, Circuitos Elétricos, Ilhas de Calor etc, fazem parte das ementas de cursos consagrados. No capítulo 5 poderão ser visto os trabalhos futuros que poderão ser feitos para avançar nesta pesquisa, a conclusão do trabalho e os resultados obtidos com ela. Pretende-se apresentar a importância do EAD e como essa modalidade educacional poderá expandir cada vez mais.

Capítulo 2

Linguagens Reflexivas

Reflexão computacional é a capacidade de um processo computacional manipular formalmente representações de suas próprias operações e estruturas. Linguagens com suporte à reflexão computacional possibilitam ao programador observar o estado interno do programa ou alterá-lo [SMITH 1982]. O paradigma de programação caracterizado por reflexão é chamado programação reflexiva.

A utilização de reflexão computacional permite que um sistema monitore seu contexto presente e, a partir desta análise, decida o que fazer a seguir. Esta capacidade confere a ele um alto grau de flexibilidade, visto que não há mais necessidade de seguir passos definidos *a priori* [FERNANDES 2008].

A Reflexão geralmente é usada em linguagens de programação de máquinas virtuais de alto nível. As linguagens Common Lisp, Scheme etc são exemplos de linguagens que permitem a reflexão.

Programação orientada à reflexão inclui auto-verificação, auto-modificação e auto-replicação. Todavia, a ênfase desse paradigma é a modificação de programas dinamicamente, a qual pode ser determinada e executada em tempo de execução. Algumas abordagens imperativas, tais como procedural e orientada a objetos, especificam uma seqüência predeterminada de operações com as quais serão processados os dados.

Entretanto a programação orientada à reflexão pode adicionar instruções ao programa que pode ser modificado dinamicamente em tempo de execução e invocado nos seus estados modificados. Isto é, a própria arquitetura do programa pode decidir em tempo de execução

o que será aplicado baseado nos dados, serviços e operações específicas.

Veja um trecho em Scheme que apresenta um exemplo:

```
;; Without reflection
(hello)

;; With reflection
(eval '(hello))

;; With reflection via string using string I/O ports
(eval (read (open-input-string "(hello)")))
```

A linguagem reflexiva utilizada nesse trabalho será o Scheme, as razões para tal e um resumo dessa linguagem são apresentados na seção 2.1 deste capítulo.

2.1 Scheme

Scheme é uma linguagem de programação multi-paradigma que suporta programação funcional e procedural e possui caráter reflexivo. Foi criada por Guy L. Steele e Gerald Jay Sussman nos anos 1970 a partir da linguagem Lisp [BITTENCOURT 2006]. Essa linguagem possui características muito atraentes, uma delas é a possibilidade do usuário escrever programas e manipulá-los. Com isso, ele poderá criar ferramentas que ajudam na concepção e implementação de aplicações.

Scheme foi uma das primeiras linguagens de programação a incorporar lambda cálculo nos procedimentos da classe principal. Também proporciona a usabilidade de regras de escopo estático e estruturas em bloco em uma linguagem tipada. Scheme foi o primeiro grande dialeto de Lisp a distinguir procedimentos de expressões lambda e símbolos e a usar um único ambiente léxico para todas as variáveis [CASAROTTO and CHISTE 2003].

A linguagem Scheme, apesar de poderosa, caracteriza-se por ter uma sintaxe simples, com poucas regras, que não exige muito tempo de aprendizagem do programador. Para essa linguagem existem várias implementações, dentre elas: Stalin [SISKIND 2008], Chicken [CHICKEN 2008] e o Bigloo [BIGLOO 2008]. Nesse trabalho será utilizado o Bigloo.

O Bigloo é uma implementação da linguagem de programação Scheme, desenvolvida no *French IT Research Institute* - INRIA [BIGLOO 2008]. Seu objetivo é fornecer ferra-

mentas para a geração de código eficiente e diversificado que pode ter o desempenho igual a programas em C ou C++.

Bigloo contém um compilador que pode gerar código em C, bytecode em Java ou .NET. Tal como acontece com outros dialetos Lisp, o Bigloo contém um intérprete, também conhecido como `read-eval-print-loop`. Alguns aplicativos como, o HOP e Roadsend PHP foram escritos em Bigloo [CURY FILHO 2008].

2.1.1 Instalação do Bigloo

O Bigloo está disponível para *download* em [STALINGUI 2008]. Ao executar o arquivo de instalação, o mesmo irá adicionar uma variável de sistema no seu computador, `BIGLOOLIB`, e adicionará os seguintes diretórios no *Path*: `C:\WinBigloo\bin;C:\WinBigloo\gcc\bin;C:\WinBigloo\lib\3.1a`. Ao concluir essa instalação é recomendado reiniciar a máquina.

Após a instalação do Bigloo o usuário poderá digitar o comando `bigloo` no *prompt*, e o intérprete ficará em um ciclo permanente de leitura-cálculo-escrita. Observe o intérprete:

```
D:\>bigloo
------------------------------------------------------------------------------
Bigloo (3.2a)                                                        ,--^,
'a practical Scheme compiler'                                  _ ___/ /|/
Tue Jan 6 18:27:22 CET 2009                                 ,;'( )__, ) '
Inria -- Sophia Antipolis                                  ;;  //   L__.
email: bigloo@sophia.inria.fr                              '   \    /  '
url: http://www.inria.fr/mimosa/fp/Bigloo                       ^    ^
------------------------------------------------------------------------------
1:=>
```

No repositório do Google, [STALINGUI 2008], criado para divulgação dos trabalhos dessa pesquisa, está disponível para *download* o instalador do Bigloo já compilado e atualizado. Esse instalador foi desenvolvido como parte dessa pesquisa de mestrado e publicado no repositório do Google como forma de auxílio aos demais usuários da linguagem. Até o momento possui vários *downloads*, de diferentes países, divididos entre as várias versões disponibilizadas (veja, Figura 2.1). No site do SourceForge também estão disponibilizados os arquivos desse projeto e podem ser acessados através do seguinte endereço: `http://sourceforge.net/projects/lemac/`. Recentemente foi criar outro repositório no Google contendo instaladores apenas para a linguagem Scheme, o projeto foi criado

com o nome WINBIGLOO[1]. No site oficial[2] da linguagem há um link para o projeto WINBIGLOO.

É possível ainda realizar o *download* dos fontes do Bigloo no site oficial da linguagem, [BIGLOO 2008], e a partir do `README` realizar o `make` e as configurações de sistema para o funcionamento da linguagem.

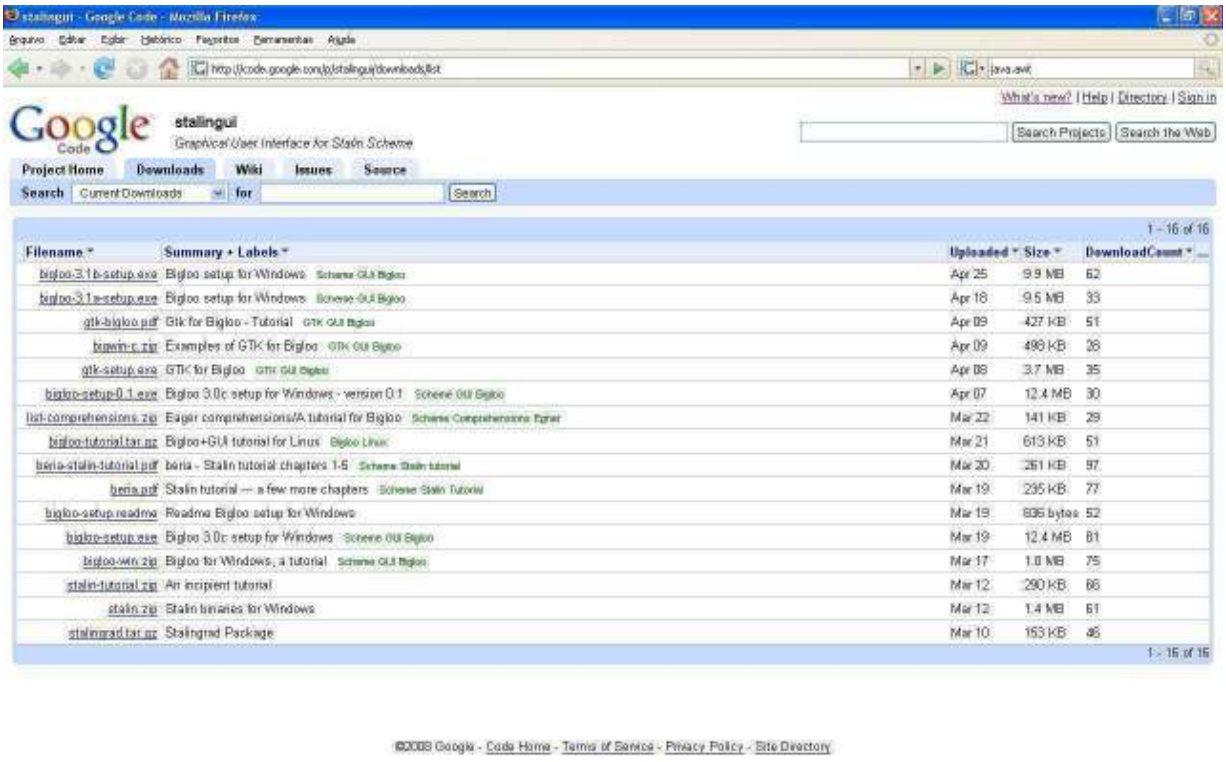

Figura 2.1: Projeto STALINGUI disponibilizado em code.google.com

A seguir são apresentadas algumas primitivas da linguagem e exemplos de algumas aplicações em Bigloo.

2.1.2 Aplicações em Bigloo

Essa linguagem disponibiliza várias funções primitivas, como, por exemplo, os operadores matemáticos +, -, ×, /. Possui ainda uma notação pré-fixada e três tipos de construções: aplicações, expressões lambda e definições.

[1]Disponível em: code.google.com/p/winbigloo
[2]BIGLOO - INRIA, disponível em: www.bigloo.org

As aplicações são listas contendo o nome de uma operação, seguidas pelos operandos, que são também chamados de argumentos. Em Scheme, assim como Lisp, as operações e os argumentos são colocados entre parênteses. Suponha uma operação de soma com os seguintes operandos (34 56 78); para tal, Scheme irá efetuar a operação e produzirá um resultado:

```
1:=> (+ 34 56 78)
168
1:=>
```

O próximo passo é adicionar produtos. Para isso, utilizam-se duas operações, multiplicação e soma. Assim, pretende-se calcular $3 \times 45 \times 63$, 4×5 e 5×7. Em Scheme, essas três operações são representadas da seguinte forma:

- $(\times\ 3\ 45\ 63)$
- $(\times\ 4\ 5)$
- $(\times\ 5\ 7)$

A fim de adicionar esses produtos, é preciso aplicar a operação de adição a eles:

```
1:=> (+ (* 3 45 63) (* 4 5) (* 5 7))
8560
1:=>
```

Invertendo as operações, considere que será calculado o produto da adição, $(3 + 45 + 63) \times (4 + 5) \times (5 + 7)$, assim teremos:

```
1:=> (* (+ 3 45 63) (+ 4 5) (+ 5 7))
11988
1:=>
```

Um exemplo com subtração e divisão pode ser visto com o cálculo da expressão: $(3 \times 4 \times 56) \div (723 - 212)$.

```
1:=> (/ (* 3 4 56) (- 732 212))
1.2923076923077
1:=>
```

Uma outra construção do Scheme a ser estudada, são as expressões lambda. Uma expressão lambda introduz variáveis para generalizar aplicações. Por exemplo, para calcular a área de um círculo de raio `r`, tem-se a fórmula πr^2. Porém para aplicar essa fórmula será necessário introduzir uma variável, r, através de uma expressão lambda.

```
1:=> (lambda(r) (* 3.1416 r r))
```

A construção lambda usada acima serve exatamente para criarmos funções. O construtor lambda atua sobre uma lista com os parâmetros da função e um corpo que calcula o valor da função. No exemplo dado, o único parâmetro é `r`, e o corpo é a expressão `(* 3.1416 r r)`. O `define` é usado para dar nome à função criada com o lambda[3]. A definição, terceira construção em Scheme, se encarrega de nomear.

```
1:=> (define area (lambda(r) (* 3.1416 r r)))
```

Uma vez definida a função, esta poderá ser usada como qualquer outra em Scheme, bastando escrever uma lista com seu nome seguido dos eventuais argumentos; após definir `area` como acima, experimente executar `(area 45)`.

De maneira geral, sempre que escrevermos uma nova função devemos testá-la, isto é, aplicá-la sobre alguns valores e verificar se ela está correta. Assim, testaremos a função no intérprete, com os seguintes raios, 45 e 8.

```
1:=> (define area (lambda(r) (* 3.1416 r r)))
area
1:=> (area 45)
6361.74
1:=> (area 8)
201.0624
```

A estrutura `define` pode ser empregada também para criar constantes e variáveis globais. Suponha, por exemplo, que o usuário deseja definir o valor de π

```
1:=> (define pi 3.1416)
pi
1:=> pi
3.1416
1:=>
```

Depois de definido o valor de π ele poderá ser utilizado para calcular a área de um círculo de raio `r`, veja:

```
1:=> (define pi 3.1416)
pi
1:=> (define (area r) (* pi r r))
area
1:=> (area 3)
28.2744
1:=>
```

[3]Lamba é o nome da letra grega λ. Essa notação é baseada no λ-cálculo, uma teoria criada na década de 30 pelo matemático Alonzo Church.

Até o momento, foi utilizado o intérprete para calcular aplicações e definir expressões lambda. O intérprete é interessante, pois possibilita ao usuário uma forma de construir suas idéias e testá-las. Porém, através da compilação é gerado um código executável, que é mais rápido e seguro. Para compilar um programa, é necessário criar um `module` que indicará ao computador onde deve ser iniciada a execução, veja o Código Fonte 2.1.

Código Fonte 2.1: Programa para calcular a área de um círculo

```
1  ;; File name: c-area.scm
2  ;; Compile:   bigloo c-area.scm -o area
3
4  (module circle (main start))
5
6  (define (start args)
7     (display "Entre com o valor do raio: ")
8     (flush-output-port (current-output-port))
9     (print (area (read)) ) )
10
11 (define pi 3.1416)
12 (define (area r)  (* pi r r))
```

As duas primeiras linhas do Código Fonte 2.1 iniciam com ponto e vírgula. Isso significa que essas linhas estão comentadas. Na primeira linha tem-se o nome do arquivo e a linha seguinte mostra como deve ser compilado o programa. A declaração do `module`

```
(module circle (main start))
```

introduz um módulo `circle` e informa que a execução começará pela função `start`.

A função `start` recebe como argumentos `args`, que contém uma lista dos elementos da linha de comando. Por exemplo, suponha que a linha de comando receba o seguinte valor:

```
area 30
```

nesse caso, `args` irá avaliar `'("area" "30")`.

Dando continuidade à avaliação do Código Fonte 2.1 a linha que contém o comando

```
(display "Entre com o valor do raio: ")
```

imprimirá seu argumento, uma string[4] ("Entre com o valor do raio: "). Entretanto, o computador pode retardar a impressão dessa string para o final da execução do

[4]String é uma sucessão de caracteres colocada entre aspas

programa. Considerando que deseja imprimir a mensagem "Entre com o valor do raio:"imediatamente, é necessário utilizar o comando:

```
(flush-output-port (current-output-port))
```

Finalmente a aplicação

```
(print (area (read)))
```

lê o raio, calcula a área correspondente e imprime o resultado com uma nova linha. A leitura é executada pela aplicação (`read`), que irá obter o valor do raio através da console.

Scheme possui ainda outros tipos de dados além de números, como por exemplo, strings.

```
"Entre com o valor do raio: "
"main"
"30"
```

Scheme também possui listas. Assim, `'("area" "30")` é um exemplo de uma lista. Existem também listas de inteiros, símbolos, string, lista vazia e outras.

Inteiros, números reais e fracionários são tipos de dados aritméticos. Há operações para controlar esses tipos de dados e elas são pouco numerosas.

$>, =, not, and, or, real?, integer?$

Além das operações para dados aritméticos existem também operações específicas para listas. Os operadores são poucos e com apenas quatro o usuário pode manipular bem esse tipo de dado, veja:

- `(car xs)` retorna o primeiro elemento de uma lista `xs`:

  `(car '(a b c))` $\rightarrow$ `a`

- `(cdr xs)` retorna a lista `xs` com o primeiro elemento removido:

  `(cdr '(a b c))` $\rightarrow$ `(b c)`

- `(cons x xs)` constrói uma nova lista, em que `car` é `x` e `cdr` é `xs`:

  `(cons 'a '(b c d))` $\rightarrow$ `(a b c d)`

- `(null? xs)` retorna `#t` se `xs` for vazio; caso contrário, retorna `#f`.

  ```
  (define xs '(a b))
  (null? xs) → #f
  (null? (cdr (cdr xs)))) → #t
  ```

Retornando ao programa do Código Fonte 2.1 observa-se que `args` contém uma lista com os elementos da linha de comando. Assim, se o usuário digita:

```
area.exe 30
```

o operando `args` receberá a lista `'("area.exe" "30")`. O segundo elemento desta lista deve indicar o raio do círculo. Nesse caso, para adquirir o raio basta utilizar `(car (cdr args))`. Entretanto, existe um problema, o valor obtido do raio é uma string "30", ao invés de um numeral 30. A linguagem Scheme permite a conversão de uma string para um número, para isso basta usar a função: `(string->number (car (cdr args)))`.

A propósito, existe uma abreviação para `(car(cdr x))`, `(cadr x)`. Usando essa abreviação o usuário pode obter o valor do raio do círculo, `(string->number (cadr args))`.

Uma outra abordagem para o cálculo da área é possível. Assim através da linha de comando, se o usuário digitar `area 30`, o programa abaixo irá produzir a área de um círculo de raio 30.

```
(module circle (main start))

(define (start args)
   (print (area (string->number (cadr args) ))))

(define pi 3.1416)

(define (area r) (* pi r r))
```

Para compilar esse programa, é necessário digitar no `prompt de comando`:

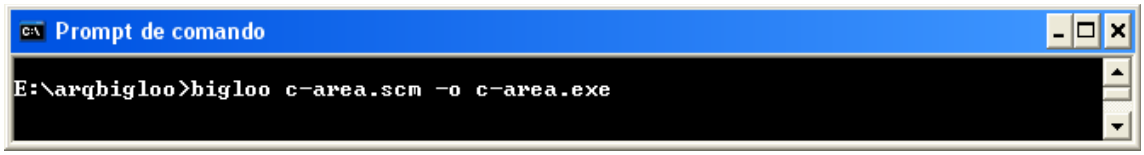

Figura 2.2: Compilando o arquivo c-area.scm

Para executar, basta:

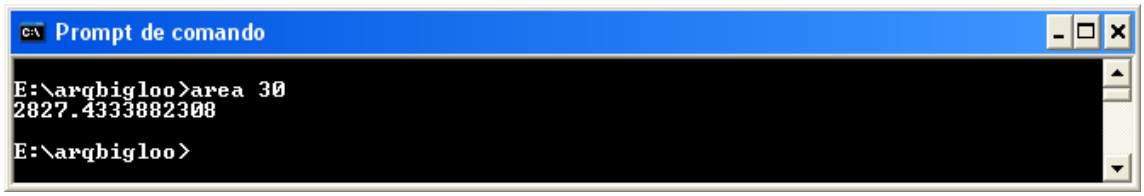

Figura 2.3: Executando o arquivo c-area.exe

O programa mostrado anteriormente apresenta um problema. Suponha que o usuário esqueça de fornecer o valor do raio. Nesse programa não há checagem dos valores passados para `args`, assim, o programa ficará aguardando por um valor do raio, que não será inserido. Uma solução é utilizar uma condição que executará o cálculo da área somente se `(cdr args)` não for nulo, veja:

```
;; File name: c-area.scm
;; Compile: bigloo c-area.scm -o area

(module circle (main start))

(define (start args)
   (when (not (null? (cdr args)))
      (area (string->number (cadr args) ))))

(define pi 3.1416)

(define (area r) (* pi r r))
```

A aplicação `(when...)` tem uma condição seguida por uma seqüência de ações, que somente são realizadas se a condição é satisfeita. Há outra opção para o `(when...)`, por

exemplo, `(cond (condições ação ...) ...)`, que possui uma lista de condições-ações e executa as ações em conjunto com a primeira condição que retorna verdadeiro. O `cond` pode ser usado para lidar com vazio `(cdr args)`, como também com argumentos não-numéricos. Na realidade, `(string->number x)` retorna `#f` se a string `x` não tiver um valor numérico equivalente. Isso conduz ao seguinte programa:

```
;; File name: c-area.scm
;; Compile: bigloo c-area.scm -o area

(module circle (main start))

(define (start args)
   (cond ((null? (cdr args)) (print "Nenhum argumento"))
         ((string->number(cadr args))
          (print (area(string->number(cadr args)))))
         (else(print "O Argumento não é um número"))))

(define pi 3.1416)
(define (area r) (* pi r r))
```

2.1.3 Interface Gráfica

Na atualidade, com o advento das Interfaces Gráficas, mudou-se o estilo dos programas. A exigência por componentes com mais usabilidade e práticos se tornou uma constante.

A linguagem Scheme pode integrar bibliotecas gráficas sem dificuldades. Nesse trabalho foi utilizada a biblioteca JAPI (*Java Application Programming Interface*) disponível para download em [STALINGUI 2008]. A razão para a escolha foi o fato de que na atualidade a maioria das máquinas já possui o JAVA, desta forma não seria necessário instalar nenhuma biblioteca adicional na máquina do usuário para executar o programa. Outras bibliotecas também poderiam ter sido utilizadas, tais como GTK desenvolvida pela `GIMP Toolkit`; exemplos de programas em Scheme utilizando a biblioteca GTK também foram feitos e estão disponíveis em [STALINGUI 2008].

Para utilização da biblioteca JAPI é necessário ter instalado na máquina o Java. Hoje, a maioria das máquinas possuem Java. Além disso são necessários:

- O arquivo `japi.sch`, que deve estar no diretório `japi`,
- Os arquivos `libjapi.a` e `japi.h`.

Todos os arquivos mencionados juntamente com o diretório `japi` devem estar no diretório onde encontram-se os fontes. O *download* desses arquivos pode ser feito em [STALINGUI 2008]. O conteúdo do arquivo `japi.sch` encontra-se disponível no Anexo B.

Depois de realizar a instalação, o usuário poderá criar uma simples GUI (Interface Gráfica do Usuário), conforme pode ser visto no Código Fonte 2.2.

Código Fonte 2.2: Uma simples GUI

```
1  ;; Compile bigloo -ljapi -lwsock32 hello.scm -o hello.exe
2
3  (module textfield
4          (include "japi/japi.sch")
5          (main start))
6
7  (define (start args)
8      (j_start)
9      (let* [ (title "Entre com o seu nome e pressione Enter")
10             (fr (j_frame title))
11             (buff (make-string 256))
12             (fld (j_textfield fr 45))
13           ]
14
15             (j_setpos fld 10 40)
16             (j_show fr)
17             (j_pack fr)
18
19             (do [(obj (j_nextaction)(j_nextaction) )
20                  (x (j_gettext fld buff)(j_gettext fld buff))]
21                 [(= obj fr)(j_quit)]
22                 (j_settext fld (string-append "Hello , " x))
23             ) ;end do
24     ) ; end let
25 ); end define
```

Examinando o Código Fonte 2.2 observa-se uma declaração desconhecida até o momento.

```
(include "japi/japi.sch")
```

O include insere no código o conteúdo de `"japi/japi.sch"`. Conforme mencionado anteriormente, esse arquivo inserido encontra-se no diretório dos fontes; desta forma, nesse diretório devem estar:

```
seu-codigo.scm
japi/japi.sch
libjapi.a
japi.h
```

A elaboração da biblioteca JAPI e os arquivos por ela utilizados não fazem parte do escopo desse trabalho. Assim, o foco nessa seção será a construção de programas utilizando a interface gráfica JAPI.

Uma outra estrutura presente no Código Fonte 2.2 é o `let`, veja-a:

```
(let [ (x (funcao-para-calcular-x))
       (y (funcao-para-calcular-y))      } ;;Variáveis Locais
       (a valor-de-a)
     ]
       ;;Instruções a serem feitas dentro do let
       (comando ....)
       (comando ....)
);end let
```

Existem dois tipos de `let`; utilizando somente `let` o programador não pode usar uma variável local para calcular outra. Entretanto se o programador optar por `let*` ele poderá utilizar qualquer definição local prévia para obter o valor de uma outra variável. Considerando o exemplo do Código Fonte 2.2, observa-se que a variável local `title` foi utilizada na definição de uma outra variável, `fr`.

Nesse exemplo, (Código Fonte 2.2), o `let` criou quatro variáveis, são elas:

title Essa variável local recebe o valor de uma string com o título da janela principal. Esse título serve como orientação para o usuário do programa.

fr Essa variável aponta para o *frame* da aplicação principal. Se o usuário clicar no `X`, localizado na parte superior à direita da janela, ele encerrará a aplicação.

buff Essa é uma string que deve ser utilizada como *buffer* para entrada de dados do *textfield*.

fld Essa variável indica para o *textfield* onde o usuário entrará com um nome.

Depois de criadas as variáveis, tem-se uma janela e um campo de texto. Agora, é necessário exibir essa janela, e isso pode ser feito através do comando: `(j_show fr)`.

Até então, não foi especificado como posicionar os componentes dentro do *frame*. Esses devem ser colocados em um pacote o mais próximo possível. Para tal, basta: `(j_pack fr)`.

Para finalizar esse programa deve-se colocar o `japi-server` em *loop* até que seja pressionado no frame o botão `X` (parte superior a direita).

```
(do [ (obj (j_nextaction) (j_nextaction))
      (x (j_gettext fld buff) (j_gettext fld buff)) ]   } ;;Variáveis locais
    [(= obj fr) (j_quit)]                               ← ;;Parada
    (j_settext fld (string-append "Hello, " x)))        ← ;;Corpo loop
```

O `do-loop` é muito parecido com o `let-macro`, com poucas diferenças. Enquanto o `let-macro` é executado uma vez, o `do-loop` é executado até que uma condição de parada seja encontrada. No exemplo do Código Fonte 2.2 a condição de parada é dada pelo segundo argumento do `do-loop`.

```
[(= obj fr) (j_quit)]
```

As ações do usuário são armazenas na variável `obj`; quando essa ação é igual ao clicar no botão `X`, o programa saíra do *loop*, irá fechar a janela e executará a instrução `(j_quit)`.

O `do-loop` também difere do `let-macro` na forma de lidar com as variáveis. Enquanto o `let-macro` especifica o valor inicial para a variável local, o `do-loop` especifica o valor inicial e o passo, ou seja, mostra como a variável irá mudar a cada interação. No exemplo do Código Fonte 2.2 tem-se duas variáveis no *loop*, são elas: `obj` e `x`.

```
(do [ (obj (j_nextaction) (j_nextaction))
      (x (j_gettext fld buff) (j_gettext fld buff)) ]
    [ (= obj fr) (j_quit)]
    (j_settext fld (string-append "Hello, " x)))
```

O valor inicial de `obj` é a saída da função `(j_nextaction)`, que lê a ação do usuário; o passo da variável também é a saída de `(j_nextaction)`. Da mesma forma, a variável `x` receberá uma seqüência de strings, cada uma representando o conteúdo do campo de texto `fld`. O corpo do *loop* será executado a cada interação.

Observando o programa do Código Fonte 2.2, o mesmo poderá ser compilado através do comando:

```
bigloo -ljapi -lwsock32 hello.scm -o hello.exe
```

Em seguida, basta executar o programa. Uma amostra desse programa pode ser vista na Figura 2.4.

Figura 2.4: Exemplo de GUI com um campo de texto

Adicionando Ações a botões

Em uma interface gráfica comumente utilizamos botões, e estes executam determinadas ações quando são ativados. Nessa sub seção é criado um programa-exemplo que contém um botão o qual tem como ação ler um número de um campo de texto, calcular o fatorial e devolver o resultado nesse mesmo campo de texto.

Figura 2.5: Exemplo de GUI para calcular Fatorial

Para criar esse exemplo, será preciso criar alguns componentes; são eles: um botão para sair da aplicação, um botão para executar a ação desejada e um campo de texto para entrada de dados.

```
(exit_button (j_button fr "Exit"))     ;;Botão de saída
(factorial_button (j_button fr " ! ")) ;;Botão para calcular o fatorial
(fld (j_textfield fr 30))              ;;Campo de texto
```

Esse programa terá um botão para calcular o fatorial situado à direita do campo de texto, a intereface com o usuário deve ser semelhante à Figura 2.5.

Há muitas maneiras de construir o *layout* da Figura 2.5 utilizando a biblioteca `japi`. A forma mais simples é criar esses componentes e em seguida escolher o tamanho (`(j_setsize var)`) e a posição (`(j_setpos var)`) deles (*frame*, botões e campos de texto) de tal modo que ocupem a posição desejada. Veja um exemplo: `(j_setsize exit-button 160 20)` - nesse caso, tem-se os seguintes argumentos passados como parâmetro: o botão exit seguido das suas dimensões de tamanho, nesse caso 160 e 20; Em `(j_setpos exit-button 20 70)` tem-se o nome do botão e em seguida as dimensões horizontal e vertical que esse botão ocupará na interface gráfica apresentada ao usuário.

Os componentes gráficos utilizados no Código Fonte 2.3, como por exemplo, `j_textfield , j_button, j_setsize`, e seus respectivos argumentos podem ser vistos em detalhes no Anexo B. Esse anexo contém todos os componentes gráficos disponibilizados até o momento no JAPI seguido dos respectivos tipos de cada argumento, e o tipo de cada componente utilizado.

O campo de entrada de dados e o botão para calcular o fatorial (representado por um ponto de exclamação) devem ter a mesma coordenada `y`, mas a coordenada `x` do botão será diferente, situando-se à direita do campo de texto. A estrutura `do-loop` será semelhante à utilizada no exemplo do Código Fonte 2.2.

O código fonte para criação desse programa pode ser visto no Código Fonte 2.3.

Código Fonte 2.3: Programa para calcular o Fatorial

```
1  ;; Compile bigloo -ljapi -lwsock32 fact.scm -o fact.exe
2
3  (module fatorial
4       (include "japi/japi.sch")
5       (main x-button))
6  (define (fac x)
7       (cond ( (< x 1) 1)
8                  (else (* x (fac (- x 1))))))
9  )
10 (define (x-button args)
11      (j_start)
12      (let* [(fr (j_frame "Fatorial"))
13             (exit-button (j_button fr "Exit"))
14             (fat-button (j_button fr "␣␣␣!␣␣␣"))
15             (fld (j_textfield fr 20))
16             (buff (make-string 256))
17             (next-number (lambda ()
```

```
18 |              (string->number (j_gettext fld buff))))                    |
19 |            ]                                                            |
20 |          (j_setsize exit-button 160 20)                                 |
21 |          (j_setpos exit-button 20 70)                                   |
22 |          (j_setsize fat-button 15 20)                                   |
23 |          (j_setpos fat-button 200 40)                                   |
24 |          (j_setsize fr 256 100)                                         |
25 |          (j_setpos fld 20 40)                                           |
26 |          (j_show fr)                                                    |
27 |          (do [(act (j_nextaction) (j_nextaction))                       |
28 |               (x (next-number) (next-number)) ]                         |
29 |              [(or (= act fr) (= act exit-button)) (j_quit) ]            |
30 |               (cond (x (j_settext fld (number->string (fac x))))        |
31 |                     (else (j_settext fld "Nao e numero...")))           |
32 |)     )   )                                                             |
```

Criando um Editor de Texto

Nessa subseção será apresentada uma maneira de escrever programas de forma elegante. Observando o Código Fonte 2.4, nota-se que, em geral, só funções que devolvem algo estão dentro da definição do `let`. As funções que não devolvem um valor, como, por exemplo, `(j_start)`, vêm antes dessa estrutura.

Existem linguagens de programação com comandos restritos e sem atribuições ou efeitos colaterais. Programadores que utilizam linguagens semelhantes a essas estão habilitados a escrever códigos mais seguros que os outros. Embora Scheme possua atribuições, o programador deve evitar usar esse recurso para garantir a confiabilidade da aplicação.

No programa dessa subseção não será utilizado o `if`, mas o `cond`. A estrutura `cond` apresenta-se mais elegante para as ações dos botões, parênteses disponíveis e permite visualizar a estrutura lógica do programa de forma mais clara.

O programa exemplo dessa subseção será a construção de um simples editor de texto. Para tal, será necessário adicionar um `textarea` e organizar os componentes em uma `grid`. Iniciando o desenvolvimento desse programa, o primeiro passo é criar uma área de texto. Em vez de inserir a área de texto diretamente no *frame*, iremos colocá-la dentro de um *panel*. Deve ser criado também um menu para organizar algumas operações, tais como: salvar um arquivo e fechar o programa. É necessário também estudar os procedimentos para escrever um texto em um arquivo.

A função para escrever o conteúdo de *textarea* em um arquivo exige dois argumentos, o *frame* e o *textarea*. Inicialmente, é necessário obter o conteúdo do *textarea* (`inhalt`)

e o nome do arquivo (`filename`). Para obter o nome do arquivo, o programa usará um diálogo que realiza uma chamada ao procedimento para capturar exceções, `with-handler`. Esse procedimento é de fundamental importância pois permite detectar alguns problemas que podem impedir o funcionamento do programa. Suponha que o usuário entre com um *filename* incorreto; utilizando o `with-handler` o programa não executará nenhuma ação caso o nome do arquivo seja inválido.

```
(define (save-file fr texto)
   (let* [ (inhalt (j_gettext texto (make-string 1000)))
           (fname (make-string 256))
           (filename (j_filedialog fr "File" "." fname))
         ]
         [with-handler (lambda(exec) #f)
           (call-with-output-file filename
             (lambda(p) (display inhalt p)))
         ]
   ) ;end let
) ;end define
```

Quando o nome do arquivo é valido, o programa abre uma porta para a gravação do arquivo, cujo nome está armazenado na string `filename`. Essa porta é passada como parâmetro:

```
[with-handler
   (lambda(exc) #f)
      (call-with-output-file filename
      (lambda(p) (display inhalt p) ) )]
```

A aplicação `(display inhalt p)` escreve o conteúdo do *textarea* na porta que é automaticamente fechada ao final da operação.

Veja o código fonte 2.4 para criação do editor de texto da Figura 2.6.

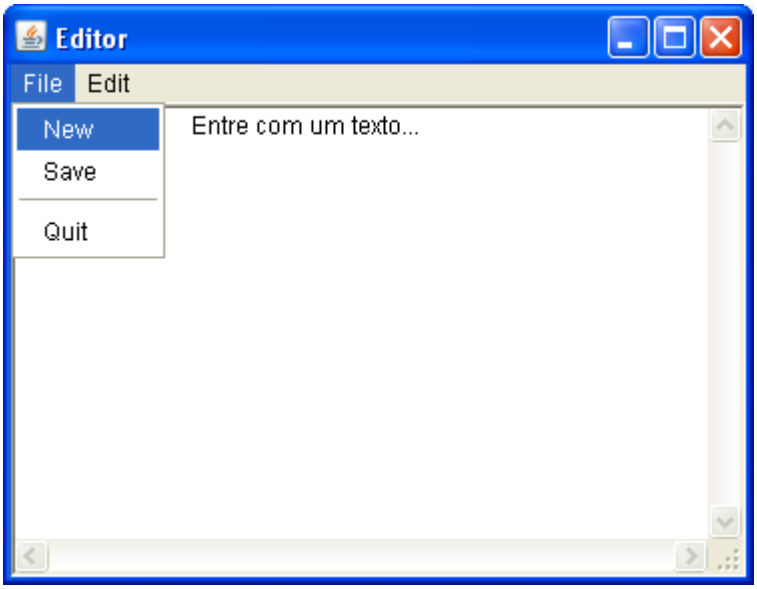

Figura 2.6: Exemplo de um Editor de Texto

Código Fonte 2.4: Programa para criar um simples Editor de Texto

```
1  (module texto
2      (include "japi/japi.sch")
3      (main main))
4
5  (define (save-file fr texto)
6     (let* [ (inhalt (j_gettext texto (make-string 1000)))
7             (fname (make-string 256))
8             (filename (j_filedialog fr "File" "." fname))
9           ]
10          [with-handler (lambda(exec) #f)
11            (call-with-output-file filename
12              (lambda(p) (display inhalt p)))
13          ]
14    ) ;end let
15 ) ;end define
16
17 (define (main args)
18    (j_start)
19    (let* [ (newtext "Entre com um texto...")
20            (fr (j_frame "Editor"))
21            (panel (j_panel fr))
22            (menubar (j_menubar fr))
23            (file (j_menu menubar "File"))
24            (new (j_menuitem file "New"))
25            (save (j_menuitem file "Save"))
26            (sep (j_seperator file))
27            (quit (j_menuitem file "Quit"))
28            (edit (j_menu menubar "Edit"))
29            (selall (j_menuitem edit "Select All"))
30            (texto (j_textarea panel 25 4))
31            (inhalt (make-string 256))
32          ]
33            (j_setgridlayout fr 1 1)
34            (j_setgridlayout panel 1 1)
```

```
35 |             (j_settext texto newtext)
36 |             (j_show fr)
37 |             (j_pack fr)
38 |
39 |             (do [(obj (j_nextaction) (j_nextaction))]
40 |                 [(or (= obj fr) (= obj quit)) (j_quit)]
41 |                 (cond ((= obj new) (j_settext texto newtext))
42 |                       ((= obj save) (save-file fr texto))
43 |                       ((= obj selall) (j_selectall texto))
44 |                 )
45 |             ) ;end do
46 |    ) ;end let
47 | ) ;end define
```

Criando uma Animação

O programa exemplo dessa subseção é desenvolver uma animação que ilustra o ciclo de Otto. O Ciclo de Otto é um ciclo termodinâmico, que idealiza o funcionamento de motores de combustão interna com ignição por faísca [IENO and NEGRO 2004].

Como pode ser visto na linha 28 do Código Fonte 2.5, esse programa em vez de utilizar `(j_nextaction)` usa `(j_getaction)`. A razão para a mudança é que o procedimento `(j_nextaction)` espera por uma ação; e esse comportamento não é apropriado para animações. Quando se desenvolve um programa com animações, elas devem ser apresentadas sem interrupções. Desta forma, o procedimento `(j_getaction)` tenta ler a ação; caso não haja nenhuma ação para executar, ele transfere a execução para o próximo comando.

O procedimento `(j_sleep 500)` realiza uma pausa de 500 milesegundos entre uma imagem e outra. Essa pausa é necessária para prevenir que os frames dessa aplicação sejam fundidos formando uma imagem borrada.

Para desenvolver esse programa é necessário conhecimentos sobre listas. Na linha 7 do Código Fonte 2.5 há uma lista com o nome das imagens que serão usadas para fazer a animação. Observa-se que essa lista possui exatamente a mesma sintaxe dos programas em Scheme.

Suponha que `xs` é uma lista que contém o nome do arquivo das imagens (animação). Nesse caso, `(car xs)` devolve o primeiro elemento de uma lista e `(cdr xs)` retorna uma outra lista, onde o primeiro elemento da lista foi removido.

Veja no quadro abaixo alguns exemplos de manipulação de listas:

xs	(define xs '("o1" "o2" "o3"))
(car xs)	"o1"
(cdr xs)	("o2" "o3")
(car(cdr xs))	"o2"
(cdr (cdr xs))	("o3")
(car(cdr(cdr xs)))	"o3"
(cdr (cdr (cdr xs)))	()

Código Fonte 2.5: Programa para criar uma Animação

```
1  ;; Compile: bigloo -o engine -ljapi -lwsock32 engine.scm
2
3  (module engine (include "japi/japi.sch")
4      (main x-animation))
5
6  (define iFiles
7    '("o1.jpg" "o2.jpg" "o3.jpg" "o4.jpg" "o5.jpg" "o6.jpg")
8  );end define
9
10 (define (x-animation argv)
11   (j_start)
12   (let* [ (fr (j_frame "Ciclo de Otto"))
13           (canvas (j_canvas fr 260 400))
14           (all-images (map j_loadimage iFiles))
15         ]
16         (j_setsize fr 270 440)
17         (j_setpos canvas 10 30)
18         (j_show fr)
19
20         (define (next images)
21            (cond [ (null? images) all-images]
22                  [ (null? (cdr images)) all-images]
23                  [else (cdr images)]
24            ) ;end cond
25         ); end define
26
27         (do [ (images  all-images  (next images))
28               (event (j_getaction) (j_getaction)) ]
29             [(= event fr) (j_quit)]
30             (j_sleep 500)
31             (j_drawimage canvas (car images) 10 0)
32         ); end do
33  );end of let*
34 );end of definition
```

A função `(next-image)`, definida nas linhas 20 a 25 do Código Fonte 2.5, faz com que, uma vez exibida uma imagem, esta seja removida da lista pela aplicação `(cdr images)` (linha 23); entretanto, se a lista estiver vazia, `(next-image)` restaura a mesma com todas as imagens (linha 22). A função `(null? images)` verifica se a lista está vazia e necessita ser restaurada.

Uma nova função apareceu no programa do Código Fonte 2.5, `map`. A função `map(fn xs)`, linha 14, aplica a função `fn` para cada elemento de uma lista `xs`, resultando uma nova lista. Utilizando o intérprete, tem-se o seguinte exemplo:

```
1:=> (define (square x) (* x x))
square
1:=> (map square '(2 3 4))
(4 9 16)
1:=>
```

Na linha 14 do Código Fonte 2.5, `(map j_loadimage iFiles)` aplica `j_loadimage` para cada elemento da lista `iFiles`, produzindo uma lista das imagens que o programa principal usará para criar a animação.

Veja o resultado do programa do Código Fonte 2.5 na Figura 2.7.

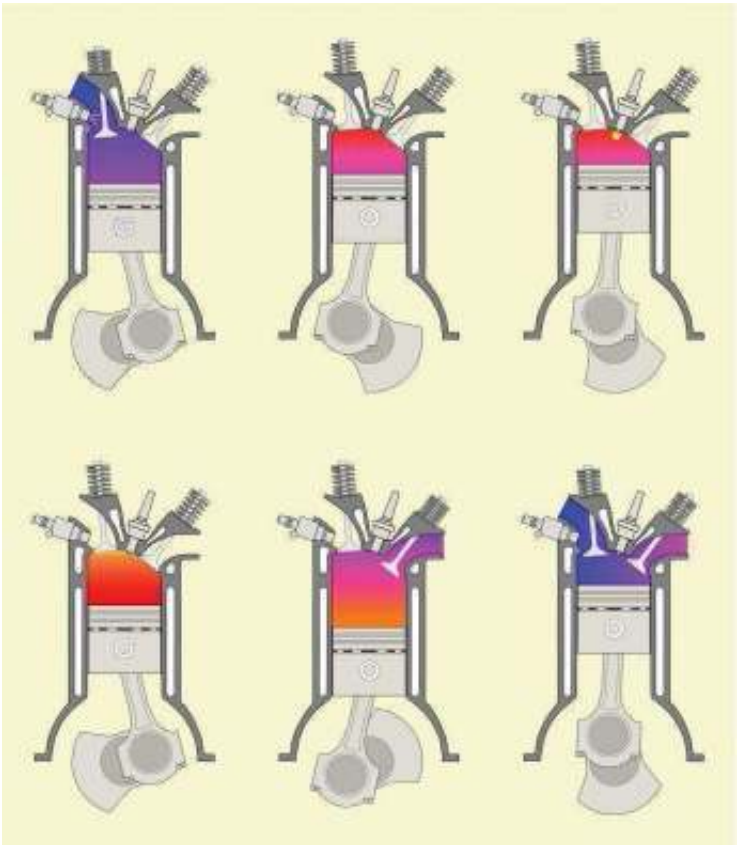

Figura 2.7: Exemplo de Animação - apresentação de uma sequência de frames que serão plotados na tela

Nesse capítulo foi feita uma breve introdução à linguagem Scheme, exemplificando a criação de programas com interfaces gráficas de caráter simples e fácil desenvolvimento. Foi apresentada a notação pré-fixada da linguagem, bem como seus operadores, tipos de dados, chamada de procedimentos e elementos gráficos.

Com essa linguagem é possível criar poderosas aplicações para os mais diversos segmentos, como, por exemplo, Educação, Medicina, Engenharia. No capítulo 4 serão apresentadas algumas aplicações que foram desenvolvidas utilizando a linguagem Scheme discutida nesse capítulo.

Capítulo 3

Materiais Auto-normalizantes para o Ensino à Distância

Material auto-normalizante é um tipo de material que pode ser utilizado em cursos à distância para auxiliar o aluno a detectar erros durante a sua aprendizagem. Ele poderá permitir que ao final do estudo o aluno tenha avançado no seu conhecimento sem as dúvidas normalmente encontradas por alunos em programas de ensino-aprendizagem onde o aluno não tem a interação necessária com o professor para sedimentar e avaliar o seu conhecimento.

O conteúdo deste capítulo faz parte de um artigo, aceito para publicação em uma enciclopédia, "*Encyclopedia of Distance Learning*", intitulado, *"Self-normalizing Distance Learning Tools"* [COSTA et al. 2008b]. Na mesma enciclopédia será publicado outro artigo, ainda no prelo, intitulado *"Participation of Distance Learning Students in Experiments"*, o conteúdo desse artigo será debatido no capítulo 4 [COSTA et al. 2008a].

Essa pesquisa está fundamentada nos estudos de biólogos evolutivos, como Dawkins, que introduziram o conceito de memes replicantes para explicar como a informação passa do professor para o estudante sem corrupção significativa. De fato, há conhecimento complexo que vai intacto de uma geração para outra, como, por exemplo, os métodos de restauração de templos de madeira japoneses, os princípios de origami, e a gramática de Sânscrito [DAWKINS 1998]. Na tentativa de preservar a informação os engenheiros de comunicação utilizam técnicas para detecção de erro, que combatem ruídos com códigos de

correção de erro que acrescentam informações extras a um sinal transmitido, aumentando a probabilidade de recuperação de dados corretos no receptor. Os projetistas de sistemas de aprendizagem online deveriam imitar esta metodologia.

Hoje, os dispositivos como telefones celulares e modems transmitem informações como uma seqüência de códigos. Durante a transmissão podem surgir ruídos, interferências que originarão erros na informação transmitida, tais erros podem ser descobertos e corrigidos. Segundo [TANENBAUM 2003], os modems detectam ruídos (erros na transmissão) utilizando algoritmos como *Checksum*, que é o módulo da soma dos dígitos de uma determinada mensagem. Se a mensagem recebida produz um *checksum* diferente do que foi transmitido, o sistema reconhece que se trata de dados errôneos. Um método simples para se calcular um *checksum* é adicionar um bit de paridade a um código que tenha 7 bits, tal que o número total de 1s seja par; se o receptor receber uma mensagem com um byte que não satisfaça esta definição, um erro está presente. Há métodos mais sofisticados de executar o *Checksum*, mas o exemplo apresentado é suficientemente adequado para mostrar a técnica.

Se as pessoas conhecessem as propriedades que impedem a informação de ser degradada, elas poderiam projetar lições mais eficientes para tutoriais online. A detecção e correção de erro não é somente possível, mas extensamente utilizada.

No ambiente educacional a detecção do erro é muito comum na relação professor-estudante. Através dos diálogos de Platão (1925) percebe-se um método inteligente de detecção de erro criado pelos gregos antigos, a dialética. Um dos truques dialéticos mais comuns é a redução por absurdo (*reduction ad absurdum*), em que o estudante tenta derivar um resultado absurdo do que ele aprendeu[1]. Em caso de sucesso, o conhecimento é marcado como corrompido. Outro método de detecção de erro comum entre os Gregos

[1]Redução por absurdo: Considere que dado uma fórmula H, se o objetivo é demonstrar que H é uma tautologia, pela redução por absurdo, suponha que H não é uma tautologia. A partir dessa suposição é utilizado um conjunto de deduções para concluir se um fato é contraditório, ou seja, um absurdo. Dessa forma se o resultado encontrado for falso significa que a suposição inicial é um absurdo. Em outras palavras, se a suposição diz que H é uma tautologia e após uma sequência de deduções é concluído um absurdo, então a afirmação "H não é uma tautologia"é um absurdo. Portanto a conclusão final é que H é uma tautotolia. Tautologia: Dizer que H é uma tautologia significa que para toda e qualquer interpretação I, I[H] = Verdadeiro.

antigos é checar o conhecimento contra fatos conhecidos.

A Lógica, uma ciência altamente desenvolvida, também criou um método para detecção de informações incorretas; Aristóteles e outros filósofos classificaram e analisaram as falácias para filtrar argumentos errados. Uma falácia é um argumento que pode ser demonstrado como falho em sua forma, o que o torna inválido no todo [TINDALE 2007].

Os logicistas[2] modernos classificam as falácias em material, verbal, e lógica. Nas Refutações Sofistas, [ARISTOTELES 1987], Aristóteles lista alguns tipos de falácias:

- Generalização que desconsidera exceções. Por exemplo: Contar mentiras é um ato desprezível, uma pessoa que conta uma mentira para alguém é considerada desprezível.

- *Argumentum ad hominem* (Argumento contra a pessoa): em vez de analisar o conteúdo da informação, a preocupação é sobre a ideologia, ou moral do autor. Por exemplo. Herrnstein e Murray (1996) discutem que a elite cognitiva está se tornando separada dos de inteligência comum. Porém a idéia dos autores deve ser falsa, uma vez que estes cientistas políticos são autores da obra *The Bell Curve*, considerada racista [HERRNSTEIN and MURRAY 1996]. Algumas vezes, *argumentum ad hominem* refere-se a moral das pessoas que têm pouco ou nada a ver com os autores do argumento refutado, um exemplo são os argumentos contra o vegetarianismo baseados no fato de que Hitler era um vegetariano.

- *Petitio Principii* (Petição de princípio) ou Argumento circular: demonstra uma conclusão por meio de premissas que assumem aquela conclusão. Por exemplo: Eu não sou marginal, porque não fiz nada criminoso.

- *Non Sequitur* (Não segue): incorretamente assume que uma coisa é a causa de outra.

- *Post hoc ergo propter hoc* (depois disto, logo causado por isto): Acreditando que sucessão temporal implica uma relação causal.

Há dois importantes fatos sobre falácias. O primeiro é que a maioria das pessoas que cometem falácias o faz sem a consciência de que o argumento deles/delas é falho

[2]Logicista é um especialista em uma subdivisão da ciência cognitiva que acredita que a lógica é suficiente para explicar qualquer problema

[COHEN 1982]. Existem padrões comuns em muitas falácias e elas podem ser facilmente perceptíveis. O segundo fato sobre falácias é que elas freqüentemente dificultam a transmissão do conhecimento; o estudante pode não entender um fato corretamente porque a lição tem um padrão enganador; um padrão falso que se encontra freqüentemente em tutoriais, um argumento circular (*Circulus in Probando*).

Falácias verbais são os erros mais comuns encontrados em lições e tutoriais; elas são descobertas facilmente por um estudante treinado. A seguir são listadas algumas falácias verbais:

- *Argumentum verbosium* (Prova por verbosidade): Autores de tutoriais acreditam que apresentar suas idéias com um volume de material possibilita que os tópicos aparentem ser bem pesquisados e fundamentados por fatos. A conseqüência é que os estudantes com todo esse volume de informação podem se desviar da informação a ser aprendida.

- Ambigüidade: consiste em empregar a mesma palavra em mais de um sentido. Para evitar este tipo de falácia, o professor deve ser cuidadoso e ter certeza de que todos os termos importantes foram definidos.

A falácia lógica é uma falha na estrutura de um argumento dedutivo, o que a torna inválida. Esse tipo de falácia não será largamente discutido nesse trabalho, porque o seu papel na detecção de erros é limitado. Desta forma, serão apresentados na próxima seção alguns tipos de erros existentes no processo de aprendizado e como os mesmos podem ser detectados e corrigidos.

3.1 Detecção de Erro

3.1.1 Classificação de Erros

Os sistemas de aprendizagem sempre foram baseados em detecção de erro, embora alguns professores e estudantes desconheçam isso. No ambiente educacional os erros podem ser classificados como:

- Erros claros: A instrução transmite um fragmento de informação que está completamente errado.

- Erros de comunicação: O estudante não pode entender a mensagem. Existem muitas causas para a mensagem não atingir o estudante; uma delas podem ser as deficiências nas aptidões do estudante ou a falta de conhecimento anterior.

- Divergências no processo de aprendizagem: O processo de aprendizagem toma um rumo não desejado. Por exemplo, quando um estudante está aprendendo a nadar, ele pode substituir uma braçada correta pela errada; ao final a braçada errada se torna automática.

As principais desvantagens da aprendizagem à distância em relação aos métodos de detecção de erro foram projetados para seres humanos em contato direto e agindo de forma síncrona; e não funcionam bem quando pessoas estão trabalhando em lugares distantes e com diferentes horários. Nesse trabalho, propõe-se a utilização de alguns métodos de detecção e recuperação de erro que poderão ser utilizados durante o processo de aprendizagem online.

3.1.2 Recuperação de Erros

A recuperação de erros apresenta-se como uma forma de retificação de algum conhecimento do aluno que por algum motivo apresenta-se incorreto. O aluno, através do material utilizado para estudo e das informações transmitidas pelo professor, pode em algum momento compreender alguma informação incorretamente ou até mesmo adquirir algum conhecimento incorreto. Desta forma, foram propostas no artigo [COSTA et al. 2008a] três formas de recuperação de erro, são elas: a natureza fornece a chave, *feedback* e balizas lingüísticas.

A Natureza fornece a chave

Sempre que o assunto em estudo trata de fenômenos naturais, uma proposta é a criação de kits e experimentos bem concebidos para trabalhos de laboratório que permitirão ao aluno:

- preencher as lacunas existentes em tutoriais online;
- descobrir erros em partes da informação;
- estabelecer pontos de controle no seu gráfico de aprendizado a fim de verificar se ele está atingindo o objetivo esperado.

Uma pessoa não aprende a cozinhar se não tenta recriar as receitas aprendidas de programas de televisão ou vídeos online. Dawkins(1998) acredita que a Natureza fornece pontos de verificação para a aquisição do conhecimento e isto é feito de forma digital, isto é, verdadeiro ou falso; para fazer um breve resumo, os pontos de verificação de Dawkins são partes do conhecimento que obedecem as três leis:

- Bivalência: Qualquer proposição é verdadeira ou falsa.
- Terceiro excluído: Uma proposição é verdade ou sua negação é verdade.
- Não-contradição: Uma proposição não pode ser ao mesmo tempo verdadeira e falsa.

Os Programas de Educação Online podem oferecer uma chave fornecida pela Natureza. Um exemplo são as pesquisas feitas pelo Dr. Kelley, que de seu escritório em Ithaca/Nova Iorque controla os dados postados no repositório por seus alunos sobre Aurora Boreal [KELLEY 1989]. É preciso ressaltar que, importantes experimentos científicos estão baseados nesta premissa. Além do mais esse trabalho possibilita a realização de experiência prática. Sem dúvida esse tipo de exercício permite ao estudante descobrir erros presentes em tutoriais, podendo o professor participar e prover uma melhoria do tutorial.

Alguns programas de graduação à distância encontraram uma solução muito simples e barata para o problema de oferecer soluções fornecidas pela Natureza a seus estudantes: esses programas oferecem créditos de residência para experiências práticas. Naturalmente, esta solução não é satisfatória para os estudantes que não têm uma experiência prévia do campo de estudo.

Feedback (Realimentação)

Mecanismos para avaliar a evolução do processo de aprendizagem podem fornecer *feedback* para o sistema que alimenta/insere a lição, para corrigir erros, reforçar conceitos, encontrar

caminhos alternativos para a aquisição de conhecimento, e redirecionar comportamentos que divergem dos objetivos propostos.

O *feedback* não precisa ser obtido de um único estudante; pode-se também obter um *feedback* útil relacionando a evolução de um estudante com outro. Uma baixa correlação pode indicar que o tutorial é ineficaz. Por exemplo, se um estudante aprende a tocar bem piano pela Internet, outro estudante não pode nem sequer tocar a melodia mais simples, e ainda um terceiro estudante fica numa posição intermediária, pode-se suspeitar, (através da correlação dos estudantes), que eles estejam exibindo conhecimentos adquiridos de experiências anteriores com a música e que o tutorial pode não ser eficaz.

Um outro mecanismo de *feedback* existente e bastante conhecido são aplicados em Sistemas Tutores Inteligentes (STI). Os STI são programas de computadores com propósitos educacionais e que incorporam técnicas de Inteligência Artificial [FOWLER 1991]. Nesses sistemas o termo inteligente representa a capacidade de resolver os problemas que apresenta-se para os alunos e explicar como foi feito. Ao contrário de métodos tradicionais os STI permitem um maior grau de individualização na instrução, em particular, um STI relaciona a instrução com o entendimento das metas e crenças dos alunos. Em [GAMBOA and FRED 2001] é possível encontrar maiores informações sobre os STI.

Balizas lingüísticas

Um conjunto bem projetado de mecanismos lingüísticos pode sinalizar se o conhecimento é corrompido, ou se o estudante esqueceu parte da informação. A poesia de Hesíodo [3][4](2007)

[3]A obra de Hesíodo, Os Trabalhos e os Dias, é caracterizada pelo uso de dialeto iônico e os versos hexâmetros dactílico característicos da epopéia.

[4]Os poemas épicos caracterizam-se por utilizar como métrica os versos hexâmetro dactílico. Para os gregos e outros povos da Antiguidade, um poema não era bem o que hoje chamamos de "poesia", e sim um arranjo de sílabas longas e breves que dava aos poemas, quando declamados, uma musicalidade característica. Os poemas eram organizados em versos, unidades rítmicas padronizadas que com o advento da escrita foram convencionalmente dispostas de modo a ocupar uma linha do texto. Não havia rimas, costume que se desenvolveu muito mais tarde com o advento do Cristianismo e da poesia religiosa. A poesia grega usava vários metros, combinações de sílabas longas e breves, e aos grupos silábicos que formavam a estrutura básica dos versos chamamos pés. Cada pé era constituído pela sucessão de longas e breves e comportava dois "tempos", um mais elevado e um mais baixo. A sucessão padronizada de pés emprestava ao verso um ritmo característico, lento e solene, ou vivaz e agitado, e assim por diante.

é um bom exemplo de baliza lingüística que previne o conhecimento de ser esquecido, devido à estrutura (dialeto, versos) em que a obra foi escrita. A poesia de Hesíodo na obra Os trabalhos e Os Dias, através de sua estrutura auxiliava a memória a detectar erros e relembrar a data correta do plantio. As balizas não precisam mostrar um caráter puramente lingüístico. Por exemplo, um estudante pode apresentar uma lição como uma sucessão de passos em uma prova associada a um sistema formal. Os teoremas intermediários representarão o papel de balizas; se o conhecimento estiver corrompido, o estudante poderá descobrir isto, desde que, utilizando o conceito montado, não demonstre a baliza.

Veja a seguir um exemplo de utilização de baliza lingüística no ensino de trigonometria. Suponha que um aluno esteja estudando sobre conceitos de operações com senos e cossenos. Uma das lições estudada apresenta o conceito das seguintes fórmulas:

$$sin(\alpha + \beta) = sin(\alpha)cos(\beta) + sin(\beta)cos(\alpha)$$
$$sin(\alpha - \beta) = sin(\alpha)cos(\beta) - sin(\beta)cos(\alpha)$$
$$cos(\alpha + \beta) = cos(\alpha)cos(\beta) - sin(\alpha)sin(\beta)$$
$$cos(\alpha - \beta) = cos(\alpha)cos(\beta) + sin(\alpha)sin(\beta)$$

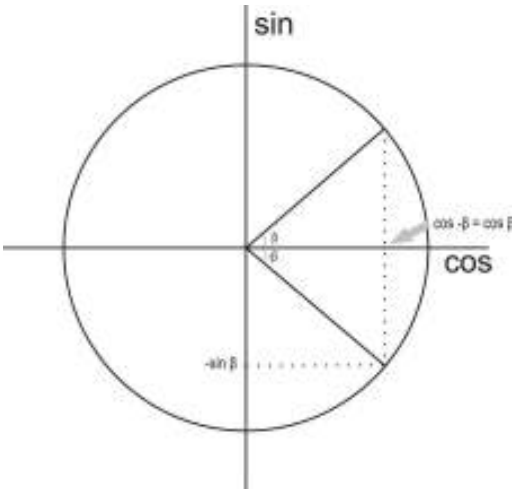

Figura 3.1: Cosseno de um ângulo negativo

Qualquer aluno poderia aprender a demonstrar estas fórmulas; porém a demonstração é um pouco complexa e leva-se muito tempo para completar a prova. Porém, um estudante pode selecionar alguns teoremas para representar o papel de balizas e encontrar uma prova alternativa e rápida para eles, isto é, uma prova que não requeira essas fórmulas. Para

construir um conjunto de provas que ajude o estudante a aprender essas fórmulas, pode ser utilizado o conceito da figura 3.1 como baliza.

A descoberta do erro pode ser realizada se o estudante comparar uma prova que usa a fórmula, com outra que não a usa. De acordo com a Figura 3.1 conclui-se que $cos(-\beta) = cos(\beta)$. Suponha que após estudar a lição o estudante acredite que a fórmula para $cos(\alpha - \beta)$ é dada abaixo:

$$cos(\alpha - \beta) = sin(\alpha)cos(\beta) - sin(\beta)cos(\alpha)$$

Ele poderá usar a fórmula acima para provar que $cos(-\beta) = cos(\beta)$, como pode ser visto a seguir. Neste caso $\alpha = 0$, substituindo nas fórmulas, tem-se:

$$\begin{gathered} cos(-\beta) = cos(0 - \beta) = sin(0)cos(\beta) - sin(\beta)cos(0) = \\ = 0 \times cos(\beta) - sin(\beta) \times 1 = \\ = -sin(\beta) \end{gathered}$$

O resultado final está obviamente errado; então a fórmula está errada e requer correção. A partir dessa observação o estudante detecta uma falha no seu conhecimento e tem a oportunidade de corrigi-lo, evitando que o erro permaneça e se dissemine.

Esse aluno mesmo distante do professor é capaz de estudar novamente o assunto e obter fontes alternativas de aquisição do conhecimento, corrigindo dessa forma o conceito que ele tinha a priori de $cos(\alpha - \beta)$.

Na próxima seção serão apresentados alguns conceitos sobre Ensino à Distância, bem como algumas metodologias utilizadas baseadas no trabalho de Skinner e Papert.

3.2 Ensino à Distância

É notada a dimensão que o Ensino à Distância (EAD) tem tomado nos últimos anos no cenário nacional e internacional. O assunto tem sido bastante discutido em congressos, seminários e salas de aula. As definições para essa modalidade de ensino são dadas por vários autores, veja algumas:

De acordo com Peters (2003) o Ensino à Distância é um método de transmitir conhecimentos, habilidades e atitudes, racionalizando, mediante a aplicação da divisão do

trabalho e de princípios organizacionais, assim como o uso extensivo de meios técnicos, especialmente, para o objetivo de reproduzir material de ensino de alta qualidade, o que torna possível instruir um grande número de alunos ao mesmo tempo e onde quer que eles vivam. É uma forma industrializada de ensino e aprendizagem.

Segundo o Decreto de Lei nº 5.622, de 19 de dezembro de 2005 [BRASIL 2005], a Educação à Distância caracteriza-se como "modalidade educacional na qual a mediação didático-pedagógica nos processos de ensino e aprendizagem ocorre com a utilização de meios e tecnologias de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos".

Durkheim *apud* Silva (1987) define que "Educação é uma coisa eminentemente social, tanto pelas suas origens como pelas suas funções e que deve ser considerada mais com relação à comunidade do que ao indivíduo".

Conforme apresentado até o momento é notado que a educação é um valioso instrumento na sociedade, sendo vital para o ser humano. A EAD é bastante discutida e possui várias frentes de opinião relacionadas às metodologias adotadas. Para esse trabalho de mestrado baseado em materiais auto-normalizantes foram utilizados alguns conceitos de Skinner e Papert para a construção de uma metodologia de trabalho.

A primeira abordagem, skinneriana, é essencialmente periférica. Para Skinner é impossível explicar o que ocorre dentro da mente do indivíduo durante o processo de aprendizagem, melhor dizendo, Skinner proibe esse tipo de preocupação. Seu foco deu-se em observar o comportamento por meio das respostas dos indivíduos a estímulos, isto é, Skinner não se preocupou com os processos intermediários entre o estímulo e a resposta.

Um conceito importante que Skinner utiliza é o de reforço. Segundo Weiss (2001) o reforço é o que resulta de um acontecimento, atuando sobre o organismo, evidenciado pelo fortalecimento de uma conduta, mostrado pela freqüência com que esse organismo reage, em forma de comportamentos, a partir de estímulos do ambiente.

Assim, Skinner não enfatiza a análise de estímulos, mas o lado do reforço, sobretudo nas contingências de reforço[5]. Isso significa que em uma relação de aprendizagem, a

[5]Contigência de Reforço consiste no arranjo de uma situação para o aprendiz a qual a ocorrência de reforço é tornada contingente à ocorrência imediatamente anterior a resposta a ser aprendida [OLIVEIRA 1978].

partir das respostas do indivíduo e do reforço estabelecido para essas respostas, é que será analisada a probabilidade de uma resposta ocorrer novamente e, assim, permite o controle do comportamento.

Skinner contribuiu também com a Instrução Programada, que foi uma das primeiras abordagens do uso do computador aplicado à Educação. Skinner acreditava que o professor sozinho não tinha condições de dar reforço a todos os alunos ao mesmo tempo. Doravante a introdução de instrumentos mecânicos com a função de auxiliar a atividade de reforço se faziam necessários e permitiriam ao professor verificar através das respostas do aluno se ele havia conseguido absorve a informação transmitida. Com essa abordagem percebe-se que a resposta do aluno é reflexo do seu entendimento acerca do assunto estudado.

Skinner fez ainda muitas sugestões interessantes para criar esquemas de detecção de erros [SKINNER 1958]. Seu método está baseado em reforço positivo entregue para recompensar pequenos passos no processo de aprendizagem; o estudante é submetido a perguntas e exercícios simples que verificam se ele está pronto para progredir numa seqüência de passos. Skinner foi também um dos primeiros pesquisadores a propor o uso de dispositivos mecânicos para impulsionar o ambiente de aprendizagem. Veja o que ele diz:

> Se o professor quer tirar proveito dos recentes avanços em estudos de aprendizagem, ele deve ter a ajuda de dispositivos mecânicos. O reforço imediato do dispositivo deve prover a resposta certa. A simples manipulação do dispositivo provavelmente estará reforçando o bastante para manter o estudante médio em trabalho por um período apropriado a cada dia. Um professor pode supervisionar uma classe inteira trabalhando ao mesmo tempo em tais dispositivos, contudo cada criança pode progredir de acordo com o seu próprio progresso e pode resolver tantos problemas quanto possível dentro do período da classe.
>
> [SKINNER 1958]

Skinner não estava pensando em ensino à distância quando ele escreveu o texto acima. Contudo, uma pessoa, pode facilmente fornecer os passos que faltam para programar dispositivos de Skinner na Internet.

Papert também aborda esse assunto, veja [PAPERT 1994]. No MIT, Seymour Papert desenvolveu uma teoria sobre aprendizagem chamada construcionismo, construído sobre o trabalho de Jean Piaget ([PIAGET 1942], [PIAGET 1953], [PIAGET 1954]) e do filósofo holandês Luitzen Brouwer ([BROUWER 1975], [HEYTING 1956]). O ponto interessante da proposição de Papert é que desde o começo, o seu método de aprendizagem requer o trabalho prático dos estudantes; a idéia é que os estudantes podem construir o conhecimento através do trabalho individual. Se um estudante adquire um conceito errado, fatos o trariam imediatamente de volta ao caminho certo. O ambiente de ensino de Papert começou com uma simples tartaruga virtual que poderia executar desenhos em uma tela de computador; hoje em dia, os estudantes de Papert montam robôs, e sistemas de aquisição de dados. As plataformas de Papert estão entre os melhores exemplos de chaves fornecidas pela Natureza.

Apesar da crítica negativa que recebe o ensino à distância ele está aqui para ficar, desde que oferece um produto altamente necessário, que possibilitará uma oportunidade de ensino e instrução para pessoas que não podem cumprir exigências de residência para graduação. Acredita-se que os pesquisadores poderão eliminar todas as falhas originadas pela falta de contato presencial entre estudantes e professores, especialmente a dificuldade de fornecer realimentação para professores, como discutido anteriormente, através de trabalhos em laboratório.

De acordo com os exemplos vistos na seção 3.1 desse trabalho, percebe-se que o ensino à distância pode recuperar os problemas causados por erros presentes na transmissão de informação e também nos conteúdos das lições. Em muitos casos, um programa de ensino à distância bem projetado pode ter mais êxito descobrindo e corrigindo erros que o contato direto entre professor e estudante, uma vez que tutoriais online reservem um papel ativo para os estudantes. O material auto-normalizante utilizado nos cursos é elaborado com exercícios e trabalhos de laboratório com o intuito de verificar ao máximo se a informação transmitida foi absorvida de forma correta ou não.

No capítulo 4 será apresentada a importância da realização de laboratórios nos cursos à distância e como eles podem ser feitos, levando conhecimento aos estudantes e capacitando-os.

Capítulo 4

Ensino à Distância - a participação do estudante na realização de trabalhos de laboratório online e observações científicas

Esse capítulo abordará a realização de laboratórios e experimentos em cursos de ensino à distância. No capítulo 3 foi apresentada a importância desse tipo de trabalho na formação dos estudantes, principalmente nos cursos em que a prática é fundamental para a formação do profissional.

O assunto desse capítulo faz parte de um artigo, no prelo, escrito para a "*Encyclopedia of Distance Learning*", cuja Primeira Edição pode ser encontrada em [HOWARD et al. 2005].

Na atualidade, a crítica mais contundente contra a aprendizagem à distância é que o trabalho em laboratório é parte essencial de todos os ramos da tecnologia e das ciências naturais. Pode-se inferir desta forma, duas observações: (a) As instituições que exigem residência fazem isso principalmente porque acreditam que criam uma oportunidade para realização de trabalhos práticos; (b) Muitos cientistas como Carl Sagan e Karl Popper pensam que observações controladas são componentes importantes do processo de aprendizagem [POPPER 2002].

Como visto, o trabalho em laboratório é parte fundamental. Nas ciências humanas o trabalho prático previne erros em projetos. Por exemplo, para que alunos possam estudar sobre o DNA é possível verificar amostras, fazer observações e muitas vezes utilizar sistemas de Coleta de Dados.

Poucos sistemas de Ensino à Distância lidam com a observação, com a prática, com a coleta de amostras e com experimentos realizados em laboratório. A falta de experiência e conhecimento prático, possibilitam críticas aos sistemas de Ensino à Distância. Por isto, muitas instituições que oferecem graduação à distância não são aceitas em muitos países.

No Brasil, por exemplo, o Conselho Federal de Biologia (CFBio) não reconheceu como biólogos os profissionais formados em cursos de biologia ou ciências biológicas ministrados à distância. A determinação do CFBio que "veta expressamente"registro desses profissionais foi publicada na resolução nº 151, de 9 de maio de 2008 no Diário Oficial da União. Veja:

> Considerando o disposto no inciso I do art. 1º da Lei Nº 6.684/79 c/c a Lei Nº 7.017/82, o qual dispõe sobre o registro de portadores de diplomas de bacharel ou licenciado em curso de História Natural, ou de Ciências Biológicas, em todas as suas especialidades ou de licenciatura em Ciências, com habilitação em Biologia; Considerando o inteiro teor do Parecer CFAP Nº 02/2008, o qual conclui pelo não reconhecimento do referido curso de Ciências Biológicas; Considerando o inteiro teor do Parecer CLN Nº 06/2008, o qual conclui pela impossibilidade de registro nos CRBios dos portadores de diplomas dos cursos EAD e de Cursos de Formação de Professores (Lei Nº 9.424, de 24/12/1996); Considerando o inteiro teor dos Pareceres CFAP Nº 01/2008 e Nº 02/2008, que dispõem sobre carga horária mínima e tempo de integralização para os cursos de Ciências Biológicas; Considerando o dever institucional do Conselho Federal de Biologia voltado à proteção da sociedade e da fiscalização do exercício profissional a teor do disposto na Lei Nº 6.684/79 c/c a Lei Nº 7.017/82; (...) resolve: Art. 1º Vetar expressamente o registro perante os Conselhos Regionais de Biologia dos portadores de diplomas dos egressos dos cursos de Educação a Distância (EAD) em Ciências Biológicas e ou Biologia e do Programa Especial de Formação Pedagógica de Docentes (Lei Nº 9.424, de 24/12/1996) [BRASIL 2008].

Segundo Wlademir João Tadei, presidente do Conselho Regional de Biologia da 1ª Região, é difícil comparar profissionais formados em cursos presenciais com os formados à distância. O desempenho do profissional em atividades de campo, pesquisa e laboratórios pode está comprometido [BASSETTE 2008]. As principais críticas aos cursos à distância são feitas devido à redução da carga horária e à ausência de trabalhos de laboratório.

Para esse trabalho foi feita uma pesquisa na *web* por sistemas de Aprendizagem à Distância oficialmente aceitos nos Estados Unidos, Canadá e Austrália. Através de consultas no *Guide to Online Schools* [1] percebe-se que a maioria das instituições encontradas limitam suas atividades aos ramos de comércio, advocacia, estudos religiosos e serviços humanos. Existem poucos cursos de Engenharia à distância, a maioria deles autorizados pela Comissão de Educação e Treinamento à Distância (*Commission of the Distance Education and Training Council* - DETC), a qual o Departamento de Educação Norte Americano reconhece como uma agência de certificação; porém os catálogos desses cursos registram apenas programas associados, ou seja, programas em que os alunos trabalharão sob a supervisão de um Engenheiro ou mestre em Engenharia.

Percebe-se que os críticos desse tipo de aprendizagem, os provedores de ensino online e as comissões de certificações concordam que as engenharias, as ciências naturais e outros cursos, em que, o trabalho em laboratório é imperativo, requerem a residência para graduação. Entende-se que alguns cursos à distância não podem fornecer a experiência prática necessária para algumas competências.

A maioria dos cientistas adota o empirismo clássico ou as orientações do racionalismo crítico de Popper em seus trabalhos. O empirismo clássico baseia-se em experimentos e observações seguidos por raciocínios indutivos. Nas palavras do filósofo e estadista inglês Francis Bacon (2007, cap. CXVII), pode-se dizer que:

[1] Disponível em: http://www.guidetoonlineschools.com/. Acessado em 10/10/2007.

> *"...via nostra ea est; ut non opera ex operibus..., sed ex experimentis causas et axiomata, atque ex causis et axiomatibus rursus nova opera et experimenta extrahamus."*
>
> Trad. "nosso método é o seguinte: Não extraímos teorias de teorias anteriores, porém de ensaios extraímos causas e axiomas, e dessas causas e axiomas, extraímos novas teorias e novos ensaios". [BACON 2007]

Popper não concorda com Bacon em opinar que a teoria científica é hipotética e é gerada pela imaginação criativa para solucionar problemas; também acredita que nenhuma atividade com resultados positivos de testes experimentais poderá confirmar uma teoria científica, já que apenas um único contra-exemplo é decisivo para mostrar que a teoria é falsa.

Popper considera a falsificabilidade como o critério único de demarcação entre o que é e o que não é ciência: uma teoria deve ser considerada científica se for falsificável. Não importa como se enfoque os trabalhos de laboratório e as observações, estes podem ser considerados a pedra angular da ciência. Popper diz ainda que "o experimento é uma reconstrução racional dos passos que levaram os cientistas a uma descoberta, ou seja, ao encontro de uma verdade nova"[POPPER 2002].

Após essas considerações pode-se concluir que o trabalho de laboratório e a prática no ensino da ciência são incontestáveis. O que se contesta através desse trabalho, é que o ensino à distância não pode fornecer o retorno indispensável que os estudantes necessitam da Natureza.

Brian D. Rude, [RUDE 1979], lista algumas razões para que seja feito o uso de experimentos em laboratório em um curso de ciência à distância; estão aqui algumas destas razões:

- para demonstrar conceitos e princípios da ciência;
- para ensinar métodos e atitudes científicas;
- para manter o assunto fundamentado na realidade.

Outra razão que Rude não contemplou está discutida por Richard Dawkins,

[DAWKINS 1998], em seu livro e examinado em detalhes por Lorenzo Matteoli em [MATTEOLI 2004]:

> Nós podemos então escrever um manual de como martelar um prego, com todos os detalhes relacionados de como pregar um prego, como a profundidade, o som, a resistência etc.. O manual... pode ser útil por reduzir o ciclo de aprendizagem, mas se você comparar a habilidade de um carpinteiro que pregue 100 pregos, mas não leu o manual, com a habilidade de outro que leu somente o manual, mas nunca golpeou um prego, você verá que o que aprendeu fazendo é muito mais capaz do que o que aprendeu somente pela leitura.

Dawkins acredita que aprender através da experiência cria um tipo de conhecimento digital, enquanto a linguagem tende a ser analógica. Por exemplo, um carpinteiro experiente fica atento às metas a serem atingidas, como: ter a cabeça do prego nivelada com a madeira. Não importa qual seja a causa para o sucesso da aprendizagem através da experiência prática, o que realmente importa é que a aprendizagem à distância não a exclua.

Outra crítica negativa recorrente sobre a aprendizagem à distância é que a exigência de residência coloca o estudante em contato com assuntos correlatos. Por exemplo, o conhecimento de Latim não é fundamental para um botânico ou advogado. Entretanto, isto é tão útil que a maioria dos botânicos e advogados adquirem esse conhecimento. Por outro lado, desde que pessoas que elaboram programas de aprendizagem à distância não estão especialmente interessadas pelos Clássicos, é provável que elas não acrescentem aulas dessas línguas em seu currículo. O resultado é um estudante com uma formação restrita.

4.1 Tutoriais Online

Atualmente, nada é mais fácil que postar um tutorial na *World Wide Web*. Todavia, vinte anos atrás, se alguém quisesse oferecer um curso de Literatura precisaria de um lugar para os estudantes se encontrarem, imprimiria o material do assunto de interesse e assim por

diante. Hoje em dia tudo o que um professor precisa para oferecer um curso é conhecer o assunto. Com a Internet o professor terá o mundo inteiro para procurar os estudantes para preencher sua classe.

Porém apesar de toda facilidade os cursos à distância ainda são alvos de inúmeras críticas. Assim, devem ser apresentadas algumas possibilidades para que esse tipo de ensino tenha mais credibilidade junto à comunidade educacional.

Uma das principais críticas a respeito desse ensino é a ausência do contato físico entre o aluno, professor e grupo escolar. Na atualidade, as tecnologias conseguem suprir essa ausência através de ambientes de interação como chats, *webcams*, fórum. Além do mais, existem modernas bibliotecas virtuais que são fontes de disseminação do conhecimento e são de fácil acesso a qualquer usuário. Existem algumas modalidades de ensino, como dança e artes marciais, em que é necessário o contato físico entre os participantes o que não é possível à distância [RUDE 1979].

Outra crítica é que os alunos necessitam realizar aulas práticas, trabalhos de laboratórios e ensaios. Esse argumento torna-se enfraquecido, já que podemos realizar experimentos remotos utilizando computadores [RUDE 1979]. Os trabalhos de laboratório e sistemas de aquisição de dados podem ser executados com vantagens utilizando um computador. Esses resultados podem alimentar um repositório na *Web* e um professor pode então usar o repositório para verificar o trabalho dos estudantes ou até mesmo para realizar um trabalho científico.

Existem duas abordagens básicas para equipar laboratórios de aprendizagem à distância.

1ª abordagem - Experimentação Remota Os estudantes têm disponíveis robôs controlados remotamente, veja [COHEN et al. 2002]; um exemplo é a Cristalografia Macromolecular da Universidade de Stanford. Os procedimentos para utilização das instalações do laboratório da Universidade de Stanford estão disponíveis em [GONZALEZ 2007] e podem ser vistos também no Anexo A. Essa abordagem faz uso de experimentos que estão distantes do aluno em um *site* da universidade. Esse laboratório trabalha com amostras enviadas por outros pesquisadores e demais usuários para testes e os robos realizam os experimentos apartir desse material.

2ª abordagem - Experimentos próximos Nessa abordagem o experimento está perto do estudante e distante do professor. Acredita-se que essa abordagem enriquece a formação do estudante, já que os estudantes que realizam somente experiências remotas em sua formação têm freqüentemente medo de controlar hardware e equipamentos. Há a venda no mercado, materiais educacionais (kits) e equipamentos K-12[2] para diversas áreas: Biologia, Química, Ciências da Terra e Físicas.

Se os estudantes, ainda jovens, começarem a lidar com hardwares de laboratório, ficarão qualificados a lidar com hardwares e materiais no mercado de trabalho. Acredita-se que a realização de experimentos projetados para estudantes é muito importante em programas de aprendizagem à distância. Veja na seção 4.2 exemplos de trabalhos de laboratório.

4.2 Aplicações: Exemplos de Trabalhos de Laboratório

4.2.1 Ilhas de Calor

Os professores de cursos à distância podem projetar trabalhos de laboratório com os quais aproveitem a localização do estudante, seu ambiente físico ou social. Veja um exemplo de trabalho de laboratório concebido à distância.

Na atualidade muitos estudantes vivem em grandes cidades. Partindo desse pressuposto poderíamos imaginar que eles não precisam de ensino à distância, já que há escolas convencionais disponíveis para eles; contudo, muitos bons colégios não estão localizados em grandes centros urbanos; além disto muitas escolas são freqüentemente perigosas em muitos lugares como Brasil, Estados Unidos, Paquistão etc.

Desta forma, é proposto um experimento para estudantes que vivem em uma grande cidade. Uma ilha de calor urbana é uma área da cidade que é mais quente do que seus arredores. Esse fenômeno afeta diretamente o bem estar dos moradores da cidade.

[2]K-12 é uma designação norte americana para a Educação Primária e Secundária

Somente nos Estados Unidos, centenas de pessoas morrem a cada ano por causa do calor extremo [MEDINA-RAMÓN 2006].

Um estudante residente na cidade certamente sentirá o fenômeno, e mostrará entusiasmo com a probabilidade de estudá-lo. Alguém poderá estudar a ilha de calor através das imagens infravermelhas prontamente disponíveis por satélite. Nesse tipo de imagem, cada cor representa uma temperatura diferente; contudo, o estudante pode construir um mapa que mostre a relação entre cor e temperatura. O estudante pode obter o mapa coletando as temperaturas em diferentes locais de sua cidade, comparando o resultado com a cor da mesma localização presente na imagem. Tecnicamente, esse tipo de observação é chamado de verdade terrestre.

Na figura 4.1, pode-se ver o circuito de um equipamento usado para coletar informações de verdade terrestre para o estudo das ilhas de calor urbanas. Esse equipamento é de fácil construção e baixo custo. Uma vez ligado ao computador do estudante, o equipamento coleta dados da verdade terrestre, o qual será enviado para a máquina do professor. O professor de posse dos resultados do trabalho dos alunos pode verificar se o mesmo está correto ou não e se o aluno conseguiu compreender a matéria.

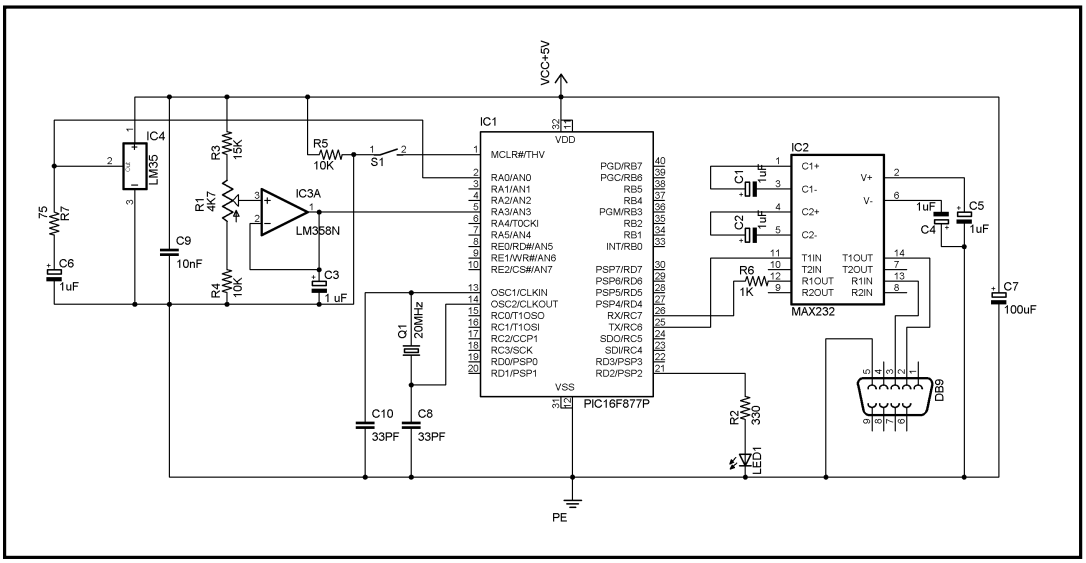

Figura 4.1: Circuito do Hardware

Conforme visto no Capítulo 2, a linguagem Scheme é bastante utilizada, principalmente em escolas dos Estados Unidos para ensinar Ciência da Computação. Assim o

aluno poderá desenvolver um algoritmo de comunicação entre duas máquinas, através de um programa em Scheme, veja a figura 4.2.

Para programar o microcontrolador do hardware desenvolvido o estudante de engenharia poderá construir um programa em C usando o compilador CSS, como pode ser visto na figura 4.3.

```
(let*
 ( (pout (open-output-file "/dev/myS0"))
  (pin (open-input-file "/dev/myS0"))
  (s (make-client-socket
        "10.23.63.139" 8080 :buffer #f))
  (p (socket-output s)))

  (write-char #\a  pout)
  (flush-output-port pout)

  (let ((ans (read-chars 19 pin)))
    (display ans p)
    (newline p)
    (flush-output-port p) )
  (read-line)
  (socket-shutdown s))
```

Figura 4.2: Algoritmo - Scheme

```
void main()
{
float valor;
int i,j;
...
pre_configuration ;
...
  while (true) {
    value = 0;
    if (((j++)%4)==0) output_high(PIN_D2);
      else output_low(PIN_D2);

      set_adc_channel(0);
      delay_us(10);
      value = read_adc()*0.2;

    delay_ms(100);
    if (kbhit()) {
      printf("Temperature = %3.2f*", valor);
      getch();
    }
  }
}
```

Figura 4.3: Programando o Microcontrolador

Esse exemplo mostra uma parte de um experimento barato, cujos resultados podem ser usados em estudos reais sobre aquecimento global. A verdade terrestre pode ser coletada por estudantes em Nairóbi, Jacarta, São Paulo, Cidade do México etc. O estudante pode construir sozinho o hardware da figura 4.1, e escrever os programas da figura 4.2 e 4.3.

Veja na figura 4.4 uma imagem do hardware que pode ser construído.

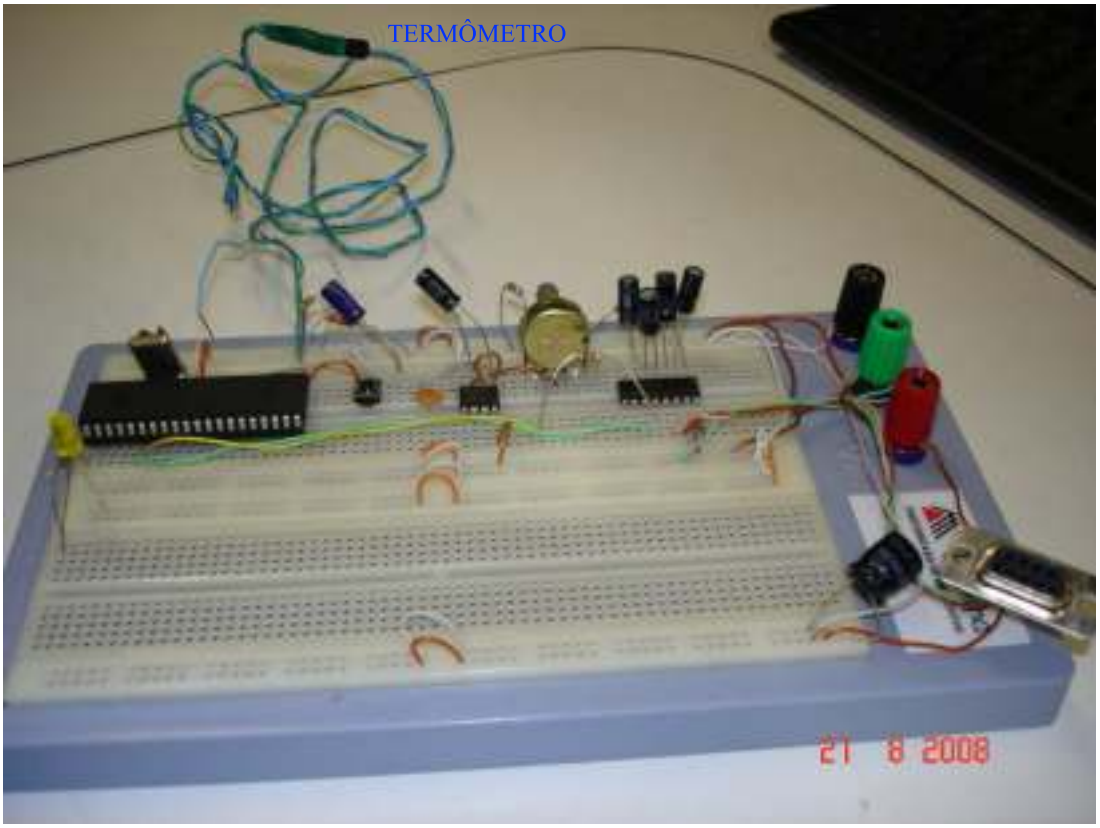

Figura 4.4: Hardware construído

Assim, percebe-se que o estudante é capaz de realizar práticas em laboratórios, construindo seus próprios experimentos e reforçando o conhecimento adquirido. A distância não é um empecilho para a disseminação do conhecimento e a tecnologia vem para proporcionar ambientes de aprendizado completos e interativos.

4.2.2 Propagação Vertical de Calor nos Oceanos

A questão ambiental, principalmente os acontecimentos decorrentes da negligência do ser humano com o Meio Ambiente estão diariamente nos noticiários. Muito tem sido discutido em salas de aula sobre as conseqüências do aquecimento global.

Com esse exemplo de trabalho pretende-se levar conhecimento aos alunos sobre a questão ambiental, ao passo que estes estarão ainda concatenados com a atualidade e poderão melhorar o próprio futuro.

Uma das conseqüências do aquecimento global é a expansão dos oceanos. Em alguns países, como a Holanda e Bangladesh, isso significa a perca de grande parte do território nacional. Nos Estados Unidos os efeitos dessa expansão já se fazem sentir; por exemplo, no estado da Luisiana, Nova Orleans já está praticamente debaixo d' água; pensa-se inclusive na transferência da cidade para outro local. Para tomar providências de proteção contra este e outros efeitos do aquecimento global é necessário modelar os fenômenos pertinentes.

Na edição da Revista "Nature"de 19/06/2008, a pesquisadora Cátia Domingues, da Csiro, Organização Nacional de Pesquisas da Austrália, afirma que até agora os cientistas têm subestimando a expansão térmica, ou seja, o aumento do volume do mar em razão do aquecimento da água.

A pesquisadora faz pela primeira vez um cálculo preciso do quanto da elevação observada no nível global dos oceanos de 1961 a 2003 pode ser atribuído a essa expansão e o quanto é culpa do derretimento das geleiras causado pelo aquecimento global.

Segundo a pesquisadora, os oceanos do planeta estão esquentando 50% mais do que se imaginava até agora e isso pode fazer com que as previsões sobre o aumento do nível do mar no fim deste século fiquem mais próximas do pior cenário.

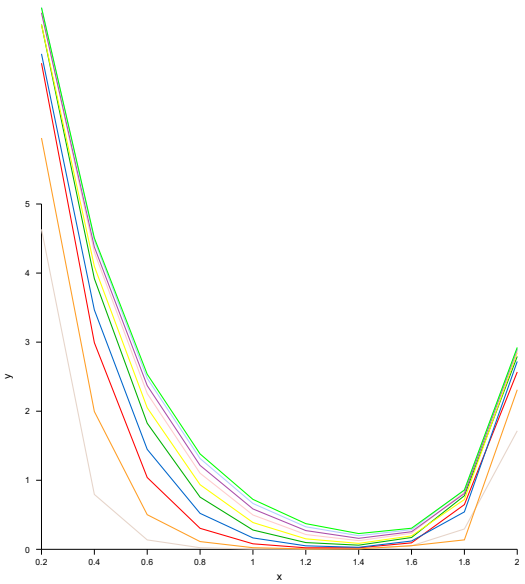

Figura 4.5: Propagação de Calor nos Oceanos

Os efeitos do aquecimento global são um assunto constante na imprensa, nas escolas e em toda a sociedade. Esse tema é abordado em inúmeros cursos e pode possuir um

caráter auto-normalizante.

Considere-se um aluno de Métodos Numéricos que realiza um trabalho para verificar a variação da temperatura nos oceanos. Para tal, é necessário o conhecimento de Equações Diferenciais Parciais, o conhecimento da Equação de Calor e a adoção de um método para solucionar tal problema.

Um aluno pode construir um programa semelhante ao fragmento do Código Fonte 4.1 e elaborar um gráfico como mostrado na figura 4.5. Para exemplificar esse trabalho com uma aplicação foi feito um algoritmo em Scheme e utilizado o Ploticus para geração dos gráficos. Para saber mais sobre o Ploticus veja em [GRUBB 2008], e informações sobre a linguagem Scheme podem ser encontradas na seção 2.1 deste trabalho.

Para realizar esse trabalho foi escolhido o método de Crank-Nicholson. A razão para tal é que, em problemas geofísicos que envolvem escalas de tempo e espaço muito grandes, instabilidades numéricas podem provocar o crescimento espúrio da solução. Desta forma, a cada passo da integração, será necessário decompor a solução em série de Fourier e eliminar os componentes espúrios do espectro.

Para resolver uma equação diferencial parcial, o primeiro passo é utilizar aproximações para transformar a equação diferencial e suas condições de contorno em um sistema de equações algébricas.

Considere as seguinte equação diferencial:

$\frac{\Delta T}{\Delta t} = \alpha \frac{\delta^2 T}{\delta x^2}$

com as condições de contorno:

t = 0, qualquer x, então, T = Tini

x = 0, qualquer $t > 0$, então, T = Te

x = L, qualquer $t > 0$, então, T = Td

A equação para aproximação é dada por:

$A = \frac{\alpha \Delta t}{(\Delta x)^2}$

Considere os pontos "internos"do domínio, a saber i = 1, 2, 3, ..., n-2, n-1 e j = 1, 2, 3, ..., m-2, m-1. Assim, temos a Equação de Crank Nicholson dada a seguir:

$$AT_{i+1,j-1} - (1 + 2A)T_{i+1,j} + AT_{i+1,j+1} = -AT_{i,j-1} + (2A - 1)T_{i,j} - AT_{i,j+1}$$

Utilizaremos essa equação para solucionar o problema dessa seção. A equação expressa anteriormente será utilizada no Código Fonte 4.1.

Foram considerados ainda nessa aplicação valores hipotéticos para calcular a variação da temperatura nos oceanos. A Temperatura média na superfície (`te`) é de 27º e no fundo do oceano (`td`) 10º. Foi aplicada a transformação rápida de Fourier a cada passo, eliminando-se as altas freqüências.

Código Fonte 4.1: Solução de Crank-Nicholson

```
(define (main args)
     (let* [(a (make-vector 10))
            (b (make-vector 10))
            (c (make-vector 10))
            (d (make-vector 10 0))
            (t (make-vector 10 0))
            (ac (eqCalor 1 0.1 0.2))
            (it (string->number(cadr args)))
            (e1  ac) (e2 (- (+ 1 (* 2 ac))))
            (e3 ac) (arq (caddr args))
            (pout (open-output-file "arq.pl"))
          ]
        (do [ (i 0 (+ i 1))]
            [ (> i 9)]
            (vector-set! a i e1)
            (vector-set! b i e2)
            (vector-set! c i e3)
        );end do
        (do [(k 1 (+ k 1))]
            [(= k (+ it 1))]
          ;; Setando o vetor d na posição 0 e n
          (vector-set! d 0 (+ (* (- ac) te)
                              (* (- (* 2 ac) 1) (vector-ref t 0))
                              (* ( - ac) (vector-ref t 1))) )
          (vector-set! d 9 (+ (* (- ac) (vector-ref t 8))
                              (* (- (* 2 ac) 1) (vector-ref t 9))
                              (* (- ac) td) ))

          ;;Setando o vetor d na posição 1 a n-1
          (do [ (i 1 (+ i 1))
                (j 1 (+ j 1))]
              [ (= i 9)]
              (vector-set! d i (+ (* (- ac) (vector-ref t (- j 1)))
                                  (* (- (* 2 ac) 1) (vector-ref t j))
                                  (* (- ac) (vector-ref t (+ j 1))) ))
          );end do

          (let [(m (TridiagonalSolve a b c d))]
               (do [ (vr 0 (+ vr 1))
                     (i 0.2 (+ i 0.2)) ]
                   [ (> vr 9) ]
                   (write i pout) (display "␣" pout)
                   (write (vector-ref m vr) pout)
                   (newline pout)
               );end do
               (print m "\n")
               (do [(r 0 (+ r 1))]
                   [(> r 9)]
                   (vector-set! t r (vector-ref m r))
```

```
50 |                    );end do                                                    |
51 |                   (gerar-arq txt)                                              |
52 |              );let                                                             |
53 |            ); end do                                                           |
54 |            (close-output-port pout)                                           |
55 |            (system (string-append arq "\\pl.exe -gif " arq "\\arq.pl"))       |
56 |            (system (string-append arq "\\arq.gif"))                            |
57 |       );end let*                                                               |
58 |);end define                                                                    |
```

Através do gráfico apresentado na figura 4.5 percebe-se a variação de calor, principalmente comparando a linha mais interna do gráfico com a mais externa. Considerando cada linha do gráfico como a variação da temperatura nas águas em um ano, um estudante, através da figura 4.5, percebe claramente a elevação da temperatura nos últimos dez anos.

Através desse exercício o aluno consegue aplicar a teoria às situações práticas e principalmente às atualidades. O professor, através do resultado do trabalho do aluno, pode checar se este compreendeu a lição.

4.3 Perspectivas

A conectividade proporcionada pela Internet não somente aumentará o número de assuntos disponibilizados on-line, mas também criará uma tendência para que os programas de graduação oferecidos on-line ultrapassem em número os tradicionais.

Considerando que a tendência é que programas de graduação à distância se tornem cada vez mais comuns, os professores deverão desenvolver a tecnologia necessária para prover os trabalhos de laboratório para estudantes, como os exemplos apresentados no tópico 4.2 desse capítulo, não importando onde eles estejam.

Além do mais existem bastante laboratórios remotos que vão desde aplicações climatológicas a genomas. Pode-se dizer que na atualidade a maioria dos cientistas trabalha com laboratórios distantes. Um exemplo são os biólogos que utilizam repositórios de genomas, ou dados de taxonomia; veja um exemplo em [SEGAL 2008].

Os Climatologistas e astrofísicos trabalham com experiências de satélite executadas por equipamentos e pessoas de apoio. Desta forma pode-se concluir que os estudantes não terão dificuldades de aprender a lidar com as Ciências Naturais através de trabalhos

de laboratório e experimentos remotos, eliminando-se, assim, a ausência de laboratórios nos cursos à distância como foi discutido no início deste capítulo.

Capítulo 5

Conclusão

Os avanços e a disseminação do uso das tecnologias de informação e comunicação descortinam novas perspectivas para o ensino à distância via internet. Através desse trabalho é possível perceber que essa modalidade de ensino cresce em nosso país e as exigências por ensino de qualidade e com oportunidades para os alunos distantes geograficamente são uma constante.

Ao desenvolver esse trabalho era esperado propor alternativas de ensino para alunos de cursos à distância. Chegando a essa etapa, observa-se que foi possível alcançar esse objetivo. Através dessa proposta, os alunos poderão, à distância, rever o seu conhecimento, verificar como está seu aprendizado e, através do material utilizado, avançar. Nesse ponto, muitos alunos de cursos presenciais poderão apresentar as mesmas dificuldades e, dependendo do material utilizado, levarão à frente conceitos errôneos.

Com os trabalhos-exemplos apresentados é perceptível os benefícios que a utilização de materiais auto-normalizantes e os trabalhos de laboratório online podem propiciar na formação dos estudantes.

Os objetivos de trabalho propostos nesse trabalho foram levantados e discutidos nos capítulos, a partir de então conclui-se.

1º objetivo - Apresentar uma proposta de material que pode ser usada em cursos à distância: Uma exemplo de atividades para compor material para cursos à distância foi apresentado no capítulo 3. Como foi apresentado através do exemplo, o estudante pode consolidar seus conhecimentos, evitando que equívocos

no aprendizado aconteça. No exemplo apresentado nesse trabalho para exercícios de trigonometria, foi possível verificar que, através de observações de conceitos correlatados e utilizando um desses conceitos como baliza, o estudante pode concluir que o seu conhecimento a respeito de $cos(\alpha - \beta)$ estava equivocado. Percebendo isso, pode reestruturar o seu conhecimento evitando assim, que o erro se propague. Assim, foi possível mostrar que há formas de se elaborar materiais que conduzem os alunos a verificar o seu conhecimento a respeito do objeto de estudo; representando uma forma de aprimoramento do conhecimento. Desta forma, não há motivos que desqualifique a utilização desse tipo de material nos cursos à distância. Pelo contrário, eles poderão, em muito, auxiliar os alunos.

2º objetivo - Exemplificar como pode acontecer aulas práticas em cursos à distância: A realização de atividades prática foi apresentado com exemplos no capítulo 4 desse trabalho. A abordagem de realização de laboratórios online exposta proporcionou um ambiente adequado para a realização de cursos, por apresentar ao estudante as mesmas condições de ensino oferecidas por cursos presenciais. Desta forma, percebe-se que a utilização das abordagens enunciadas, neste trabalho, poderá beneficiar o estudante propiciando-lhe uma oportunidade de ensino de qualidade, com materiais teóricos e exercícios de fixação que reforçam e verificam se o aprendizado aconteceu ou não, além da oportunidade de aliar teoria e prática.

Nos exemplos de laboratório online enunciados foi possível verificar, que qualquer aluno de graduação poderia está desenvolvendo essas atividades como parte complementar dos seus cursos. Um aluno de Métodos Numéricos estudará Equações Diferenciais Parciais e, aliado às informações nos noticiários sobre alterações climáticas, poderá se entusiasmar em simular a variação do nível das águas nos oceanos. Esse tipo de atividade poderá ser desenvolvido à distância e os resultados poderão ser acompanhados por um professor. A mesma atividade também poderá ser proposta para alunos de cursos presenciais, não evidenciando diferença entre essas modalidades.

O trabalho-exemplo apresentado no capítulo 4 para construção de um termômetro que, ligado a um computador envia dados sobre a temperatura de um ambiente

para a máquina de um professor representa uma possibilidade de atividade prática que pode ser concebida a distância. Assim, se o motivo que descredencia os cursos de engenharia à distância é a ausência de trabalhos de laboratório, através desse trabalho foi apresentada que há possibilidades de realização desse tipo de atividades, inclusive em cursos de engenharia.

3º objetivo - Discutir a importância de atividades de laboratório: A importância dessas atividades em cursos à distância foi abordado no capítulo 4, apresentando citações de Popper e Bacon que concordam que os trabalhos de laboratório e a prática da ciência são incontestáveis para a formação dos alunos. Com o exemplo mostrado neste trabalho foi visto que é possível inserir essas atividades no currículo de disciplinas à distância. E, os alunos, com as informações teóricas estudadas terão condições de desenvolver as atividades práticas tão importantes para a sua formação.

4º objetivo - Apresentar conceitos iniciais de uma linguagem de programação: No capítulo 2 foi apresentado conceitos sobre a linguagem de programação Bigloo, mostrando para o leitor informações desde a estrutura básica da linguagem a construção de programas-exemplos utilizando uma interface gráfica. Foi apresentado ainda programas em Bigloo para exemplificar as propostas de trabalho.

Nesse momento percebe-se que as aulas práticas, exigências de vários cursos, poderão ser realizadas à distância sem perda de qualidade, através de laboratórios online. Na atualidade, como foi apresentado nesse trabalho, muitos cientistas já trabalham com esse conceito reunindo pesquisadores de diversos locais.

As críticas ao EAD levantadas no início desse trabalho são minimizadas à medida que é possível acompanhar o desenvolvimento do aluno e verificar se houve aprendizagem através do uso de materiais auto-normalizantes. É possível ainda, levar trabalhos práticos e observações científicas para o âmbito de cursos de graduação à distância, consolidando o aprendizado do aluno e qualificando-o para o mercado de trabalho. Desta forma, foi alcançado o objetivo desse trabalho ao minimizar as críticas envolvendo o EAD, apresentado alternativas para efetivá-lo.

A perspectiva para o EAD no Brasil é de ampla expansão, acelerado ainda mais pela Internet. Muitas instituições optarão por parte da sua carga horária à distância e muitos

outros cursos serão abertos somente à distância. Acredita-se que essa modalidade de ensino poderá levar o ensino a uma democratização acentuada, permitindo o acesso à educação e à formação a uma grande parcela da sociedade.

Como atividade futura, pretende-se dar continuidade a essa pesquisa, desenvolvendo materiais para cursos e apresentando a comunidade outras formas de se fazer laboratórios online.

Referências Bibliográficas

[ARISTOTELES 1987] ARISTOTELES (1987). *On Sophistical Refutations*. Loeb Classical Library.

[BACON 2007] BACON, F. (2007). *Aphorismi de Interpretatione Naturae et Regno Hominis. In Novum Organum. (CXVII).* Disponível em: http://www.thelatinlibrary.com/bacon/bacon.liber1.shtml.

[BASSETTE 2008] BASSETTE, F. (2008). Conselho de biologia nega registro para formados à distância. *G1 - O Portal de Notícias da Globo*.

[BIGLOO 2008] BIGLOO (2008). *Bigloo Homepage. Apresenta documentação, downloads, licenças e bibliotecas adicionais do Bigloo.* Desenvolvida pela Inria Sophia-Antipolis. Disponível em: http://www-sop.inria.fr/mimosa/fp/Bigloo/. Acessado em: 6 maio 2008.

[BITTENCOURT 2006] BITTENCOURT, G. (2006). *Inteligência Artificial: ferramentas e teorias*. Editora da UFSC, Florianópolis.

[BRASIL 2005] BRASIL (19 de dezembro de 2005). Decreto de lei n° 5.622, de 19/12/2005. estabelece as diretrizes e bases da educação nacional. In *Diario Oficial [da] Republica Federativa do Brasil*. Brasília.

[BRASIL 2008] BRASIL (9 de maio de 2008). Resolução n° 151, de 09/05/2008. impossibilita o registro nos crbios de portadores de diplomas dos cursos de educação à distância. In *Diário Oficial da União. Seção 1*, page 59. Brasília.

[BROUWER 1975] BROUWER, L. E. J. (1975). *Collected Works 1. Philosophy and Foundations of Mathematics*. North-Holland, Amsterdam.

[CASAROTTO and CHISTE 2003] CASAROTTO, D. C. and CHISTE, J. G. S. (2003). *A Linguagem Scheme*. Departamento de Informática e Estatística (INE), Universidade Federal de Santa Catarina (UFSC).

[CHICKEN 2008] CHICKEN (2008). *The Chicken Wiki. Apresenta a documentação para o Chicken.* Disponível em: http://chicken.wiki.br/index. Acessado em: 6 maio 2008.

[COHEN et al. 2002] COHEN, A. E., ELLIS, P. J., MILLER, M. D., DEACON, A. M., and PHISACKERLEY, R. P. (2002). An automated system to mount cryo-cooled protein crystals on a synchroton beamline, using compact samples cassettes and a small scale robot. *Journal of Applied Chrystallography*, (35):720–726.

[COHEN 1982] COHEN, L. J. (out. 1982). Are people programmed to commit fallacies? further thoughts about the interpretation of experimental data on probability judgment. In *Journal for the Theory of Social Behaviour*, volume 12, pages 251–274.

[COSTA et al. 2008a] COSTA, E., BARBAR, J. S., CURY, R., and ROCHA, J. M. (2008a). Participation of distance learning students in experiments. In HOWARD, C., BOETTCHER, J., JUSTICE, L., ROGERS, P., and BERG, G., editors, *Encyclopedia of Distance Learning*. Idea Group Publishing. (No prelo), 2ª edition.

[COSTA et al. 2008b] COSTA, E., CURY, R., and ROCHA, J. M. (2008b). Self-normalizing distance learning tools. In HOWARD, C., BOETTCHER, J., JUSTICE, L., ROGERS, P., and BERG, G., editors, *Encyclopedia of Distance Learning*. Idea Group Publishing. (No prelo), 2ª edition.

[CURY FILHO 2008] CURY FILHO, R. (2008). Interfaces gráficas referencialmente claras e sua utilização na criação de laboratórios para o ensino a distância on-line. Tese (doutorado), Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Uberlândia, Uberlândia.

[DAWKINS 1998] DAWKINS, R. (1998). *Unweaving the rainbow.* Houghton Mifflin, Boston.

[ERIKSSON 2005] ERIKSSON, T. (2005). Remote unix desktop. disponível em: http://smb.slac.stanford.edu/facilities/remote_access/. acessado em: 09/10/2007.

[FERNANDES 2008] FERNANDES, A. P. (2008). Reflexão computacional. centro de ciências da economia e informática e do centro de ciências da saúde (urcamp).

[FOWLER 1991] FOWLER, D. (1991). A model for design intelligent tutoring systems. *Journal of Medical Systems. Vol. 15, n.1.*

[GAMBOA and FRED 2001] GAMBOA, H. and FRED, A. (2001). Designing intelligent tutoring systems: a bayesian approach. *International Conference on Enterprise Information Systems - ICEIS 2001.*

[GONZALEZ 2007] GONZALEZ, A. (2007). Remote experiments. disponível em: http://smb.slac.stanford.edu/users_guide/manual/remote_experiments.html. acessado em: 26/09/2007.

[GRUBB 2008] GRUBB, S. (2008). *Ploticus*. Um pacote de software, GPL, não-interativo para produção de plots, quadros e gráficos de dados.

[HERRNSTEIN and MURRAY 1996] HERRNSTEIN, R. J. and MURRAY, C. (1996). *The Bell Curve*. Free Press.

[HEYTING 1956] HEYTING, A. (1956). Intuitionism, an introduction. In *Journal Symbolic Logic*, volume 21, pages 367–371. North-Holland.

[HOWARD et al. 2005] HOWARD, C., BOETTCHER, J., JUSTICE, L., ROGERS, P., and BERG, G., editors (2005). *Encyclopedia of Distance Learning*. Idea Group Publishing. First Edition.

[IENO and NEGRO 2004] IENO, G. and NEGRO, L. (2004). *Termodinâmica*, volume 1. Prentice-Hall.

[KELLEY 1989] KELLEY, M. C. (1989). The earth's ionosphere: Plasma physics and electrodynamics. volume 43. International Geophysics Series, San Diego.

[MATTEOLI 2004] MATTEOLI, L. (2004). *Complexity, Chaos and Knowledge Management*. Disponível em: http://members.iinet.net.au/ matteoli/html/Articles/complexity.html. Acessado em: 10/10/2007, Scarborough.

[MEDINA-RAMÓN 2006] MEDINA-RAMÓN, M., Z. A. C. D. P. S. J. (2006). Extreme temperatures and mortality: Assessing effect modification by personal characteristics and specific cause of death in a multi-city case-only analysis. In *Environmental Health Perspect*, volume 114, pages 1331–1336.

[OLIVEIRA 1978] OLIVEIRA, J. A. (1978). *Tecnologia Educacional: Teorias da Instrução*. Vozes, Petrópolis, 6 edition.

[PAPERT 1994] PAPERT, S. (1994). *The Children's Machine: Rethinking School in the Age of the Computer*. Basic Books.

[PIAGET 1942] PIAGET, J. (1942). *Classes, Relations et Nombre*. Librairie Philosophique J. Vrin, Sorbone.

[PIAGET 1953] PIAGET, J. (1953). *The origins of Intelligence in Children*. Routledge and Kegan Paul, London.

[PIAGET 1954] PIAGET, J. (1954). *The Construction of reality in Child*. Basic Books.

[POPPER 2002] POPPER, K. R. (2002). *The Logic of Scientific Discovery*. Routledge Classics, New York, 1ª edition.

[RUDE 1979] RUDE, B. D. (1979). *Tactics and Strategies of Classroom Discipline*. Exposition Phoenix Press.

[SEGAL 2008] SEGAL, E. (2008). Segal lab of computational biology. disponível em: http://genie.weizmann.ac.il/index.html.

[SISKIND 2008] SISKIND, J. M. (2008). *STALIN - Jeffrey Mark Siskind´s Software*. Disponível em: http://cobweb.ecn.purdue.edu/ qobi/software.html. Acessada em: 3 julho 2008.

[SKINNER 1958] SKINNER, B. F. (1958). Teaching machines. In *Science*, page 128.

[SMITH 1982] SMITH, B. C. (1982). Reflection and semantics in a procedural language. Technical Report 272, MIT Laboratory of Computer Science.

[STALINGUI 2008] STALINGUI (2008). *STALINGUI Google Code. Repositório de Códigos Fontes do Google Code.* Desenvolvido por Junia Magalhães, Philippos Apolinário, Reny Cury, Jorlano Donizete, João Barbosa. Apresenta documentação, downloads e bibliotecas adicionais do Bigloo e Stalin. Disponível em: http://code.google.com/p/stalingui/. Acessada em: 3 junho 2008.

[TANENBAUM 2003] TANENBAUM, A. S. (2003). *Redes de Computadores.* Campus.

[TINDALE 2007] TINDALE, C. W. (2007). *Fallacies and Argument Appraisal.* Cambridge University Press.

Anexo A - Instalações de Cristalografia Macromolecular da Universidade de Stanford

Os procedimentos para utilização das instalações são listados nesse anexo e podem ser encontrados com detalhes em [GONZALEZ 2007].

Procedimentos:

- Determinar o professor ou monitor oficial, que será o intermediário durante a realização do experimento, consultando sempre que necessário a equipe de apoio;
- Faça o *download* e instale o software livre `NX client`. Verifique se o cliente `NX` pode ser utilizado com sucesso para o acesso aos computadores da `SMB` antes da realização do experimento; mais detalhes em [ERIKSSON 2005].
- Os usuários iniciantes devem realizar um experimento com algum membro da equipe de apoio antes do horário programado.
- Antes de enviar as amostras para `SSRL`, leia cuidadosamente todas as instruções. Preste atenção aos seguintes itens:
    1. Só use modelos de conectores que são compatíveis com o robô de `SSRL (SAM)`
    2. Conheça as práticas de segurança do `SSRL` para controle adequado do nitrogênio líquido.
    3. Substitua o nitrogênio líquido do recipiente de carga assim que qualquer gelo se forme.

Anexo B - Biblioteca JAPI - *Java Application Programming Interface*

No capítulo 2 foi apresentado exemplos de programas usando a biblioteca JAPI para construção de Interfaces Gráficas. Um dos arquivos necessários para utilização é o arquivo `japi.sch` encontrado no diretório `japi/`.

O conteúdo desse arquivo segue a seguir.

Código Fonte 5.1: Arquivo japi.sch

```
1  |                                                                                |
2  |(directives                                                                     |
3  | (extern                                                                        |
4  |    (include "japi.h")                                                          |
5  |    ;;                                                                          |
6  |    (type   int* (pointer int)       "int*")                                    |
7  |    ;;(type   char* (pointer char)      "char*")                                 |
8  |    ;;  BOOLEAN                                                                 |
9  |    (macro J_TRUE::int       "J_TRUE")                                          |
10 |    (macro J_FALSE::int      "J_FALSE")                                         |
11 |    ;;  ALIGNMENT                                                              |
12 |    (macro  J_LEFT::int      "J_LEFT")                                          |
13 |    (macro  J_CENTER::int        "J_CENTER")                                    |
14 |    (macro  J_RIGHT::int         "J_RIGHT")                                     |
15 |    (macro  J_TOP::int       "J_TOP")                                           |
16 |    (macro  J_BOTTOM::int        "J_BOTTOM")                                    |
17 |    (macro  J_TOPLEFT::int           "J_TOPLEFT")                               |
18 |    (macro  J_TOPRIGHT::int          "J_TOPRIGHT")                              |
19 |    (macro  J_BOTTOMLEFT::int        "J_BOTTOMLEFT")                            |
20 |    (macro  J_BOTTOMRIGHT::int       "J_BOTTOMRIGHT")                           |
21 |    ;;  CURSOR                                                                 |
22 |    (macro  J_DEFAULT_CURSOR::int    "J_DEFAULT_CURSOR")                        |
23 |    (macro  J_CROSSHAIR_CURSOR::int "J_CROSSHAIR_CURSOR")                       |
24 |    (macro  J_TEXT_CURSOR::int       "J_TEXT_CURSOR")                           |
25 |    (macro  J_WAIT_CURSOR::int       "J_WAIT_CURSOR")                           |
26 |    (macro  J_SW_RESIZE_CURSOR::int "J_SW_RESIZE_CURSOR")                       |
27 |    (macro  J_SE_RESIZE_CURSOR::int "J_SE_RESIZE_CURSOR")                       |
28 |    (macro  J_NW_RESIZE_CURSOR::int "J_NW_RESIZE_CURSOR")                       |
29 |    (macro  J_NE_RESIZE_CURSOR::int "J_NE_RESIZE_CURSOR")                       |
30 |    (macro  J_N_RESIZE_CURSOR::int  "J_N_RESIZE_CURSOR")                        |
31 |    (macro  J_S_RESIZE_CURSOR::int  "J_S_RESIZE_CURSOR")                        |
32 |    (macro  J_W_RESIZE_CURSOR::int  "J_W_RESIZE_CURSOR")                        |
33 |    (macro  J_E_RESIZE_CURSOR::int  "J_E_RESIZE_CURSOR")                        |
```

```
34 |     (macro  J_HAND_CURSOR::int       "J_HAND_CURSOR")                  |
35 |     (macro  J_MOVE_CURSOR::int       "J_MOVE_CURSOR")                  |
36 |     ;;  ORIENTATION                                                    |
37 |     (macro  J_HORIZONTAL::int        "J_HORIZONTAL")                   |
38 |     (macro  J_VERTICAL::int          "J_VERTICAL")                     |
39 |     ;;  FONTS                                                          |
40 |     (macro  J_PLAIN::int             "J_PLAIN")                        |
41 |     (macro  J_BOLD::int              "J_BOLD")                         |
42 |     (macro  J_ITALIC::int            "J_ITALIC")                       |
43 |     (macro  J_COURIER::int           "J_COURIER")                      |
44 |     (macro  J_HELVETIA::int          "J_HELVETIA")                     |
45 |     (macro  J_TIMES::int             "J_TIMES")                        |
46 |     (macro  J_DIALOGIN::int          "J_DIALOGIN")                     |
47 |     (macro  J_DIALOGOUT::int         "J_DIALOGOUT")                    |
48 |     ;;  COLORS                                                         |
49 |     (macro  J_BLACK::int             "J_BLACK")                        |
50 |         (macro  J_WHITE::int             "J_WHITE")                    |
51 |     (macro  J_RED::int               "J_RED")                          |
52 |     (macro  J_GREEN::int             "J_GREEN")                        |
53 |     (macro  J_BLUE::int              "J_BLUE")                         |
54 |     (macro  J_CYAN::int              "J_CYAN")                         |
55 |     (macro  J_MAGENTA::int           "J_MAGENTA")                      |
56 |     (macro  J_YELLOW::int            "J_YELLOW")                       |
57 |     (macro  J_ORANGE::int            "J_ORANGE")                       |
58 |     (macro  J_GREEN_YELLOW::int      "J_GREEN_YELLOW")                 |
59 |     (macro  J_GREEN_CYAN::int        "J_GREEN_CYAN")                   |
60 |     (macro  J_BLUE_CYAN::int         "J_BLUE_CYAN")                    |
61 |     (macro  J_BLUE_MAGENTA::int      "J_BLUE_MAGENTA")                 |
62 |     (macro  J_RED_MAGENTA::int       "J_RED_MAGENTA")                  |
63 |     (macro  J_DARK_GRAY::int         "J_DARK_GRAY")                    |
64 |     (macro  J_LIGHT_GRAY::int        "J_LIGHT_GRAY")                   |
65 |     (macro  J_GRAY::int              "J_GRAY")                         |
66 |     ;;  BORDERSTYLE                                                    |
67 |     (macro  J_NONE::int              "J_NONE")                         |
68 |     (macro  J_LINEDOWN::int          "J_LINEDOWN")                     |
69 |     (macro  J_LINEUP::int            "J_LINEUP")                       |
70 |     (macro  J_AREADOWN::int          "J_AREADOWN")                     |
71 |     (macro  J_AREAUP::int            "J_AREAUP")                       |
72 |     ;;  MOUSELISTENER                                                  |
73 |     (macro  J_MOVED::int             "J_MOVED")                        |
74 |     (macro  J_DRAGGED::int           "J_DRAGGED")                      |
75 |     (macro  J_PRESSED::int           "J_PRESSED")                      |
76 |     (macro  J_RELEASED::int          "J_RELEASED")                     |
77 |     (macro  J_ENTERERD::int         "J_ENTERERD")                      |
78 |     (macro  J_EXITED::int            "J_EXITED")                       |
79 |     (macro  J_DOUBLECLICK::int       "J_DOUBLECLICK")                  |
80 |     ;;  COMPONENTLISTENER                                              |
81 |     ;;  J_MOVED                                                        |
82 |     (macro  J_RESIZED::int           "J_RESIZED")                      |
83 |     (macro  J_HIDDEN::int            "J_HIDDEN")                       |
84 |     (macro  J_SHOWN::int             "J_SHOWN")                        |
85 |     ;;  WINDOWLISTENER                                                 |
86 |     (macro  J_ACTIVATED::int         "J_ACTIVATED")                    |
87 |     (macro  J_DEACTIVATED::int       "J_DEACTIVATED")                  |
88 |     (macro  J_OPENED::int            "J_OPENED")                       |
89 |     (macro  J_CLOSED::int            "J_CLOSED")                       |
90 |     (macro  J_ICONIFIED::int         "J_ICONIFIED")                    |
91 |     (macro  J_DEICONIFIED::int       "J_DEICONIFIED")                  |
92 |     (macro  J_CLOSING::int           "J_CLOSING")                      |
93 |     ;;  IMAGEFILEFORMAT                                                |
94 |     (macro  J_GIF::int               "J_GIF")                          |
95 |     (macro  J_JPG::int               "J_JPG")                          |
```

```
96 |     (macro  J_PPM::int                "J_PPM")                                  |
97 |     (macro  J_BMP::int                "J_BMP")                                  |
98 |     ;;                                                                          |
99 |     (macro j_start::int () "j_start");                                          |
100 |    (macro j_connect::int (string) "j_connect")                                  |
101 |    (macro j_setdebug::void (int) "j_setdebug")                                  |
102 |    (macro j_frame::int (string) "j_frame")                                      |
103 |    (macro j_button::int (int string) "j_button")                                |
104 |    (macro j_graphicbutton::int (int string) "j_graphicbutton")                  |
105 |    (macro j_checkbox::int (int string) "j_checkbox")                            |
106 |    (macro j_label::int (int string) "j_label")                                  |
107 |    (macro j_graphiclabel::int (int string) "j_graphiclabel")                    |
108 |    (macro j_canvas::int (int int int) "j_canvas")                               |
109 |    (macro j_panel::int (int) "j_panel")                                         |
110 |    (macro j_borderpanel::int (int int) "j_borderpanel")                         |
111 |    (macro j_radiogroup::int (int) "j_radiogroup")                               |
112 |    (macro j_radiobutton::int (int string) "j_radiobutton")                      |
113 |    (macro j_list::int (int int) "j_list")                                       |
114 |    (macro j_choice::int (int) "j_choice")                                       |
115 |    (macro j_dialog::int (int string) "j_dialog")                                |
116 |    (macro j_window::int (int) "j_window")                                       |
117 |    (macro j_popupmenu::int (int string) "")                                     |
118 |    (macro j_scrollpane::int (int) "j_scrollpane")                               |
119 |    (macro j_hscrollbar::int (int) "j_hscrollbar")                               |
120 |    (macro j_vscrollbar::int (int) "j_vscrollbar")                               |
121 |    (macro j_line::int (int int int int) "j_line")                               |
122 |    (macro j_printer::int (int) "j_printer")                                     |
123 |    (macro j_image::int (int int) "j_image")                                     |
124 |    (macro j_filedialog::string (int string string string)                       |
125 |                "j_filedialog")                                                  |
126 |    (macro j_fileselect::string (int string string string)                       |
127 |                "j_fileselect")                                                  |
128 |    (macro j_messagebox::int (int string string) "j_messagebox")                 |
129 |    (macro j_alertbox::int (int string string string) "j_alertbox")              |
130 |    (macro j_choicebox2::int (int string string string string)                   |
131 |                "j_choicebox2")                                                  |
132 |    (macro j_choicebox3::int (int string string string string string)            |
133 |                "j_choicebox3")                                                  |
134 |    (macro j_additem::void (int string) "j_additem")                             |
135 |    (macro j_textfield::int (int int) "j_textfield")                             |
136 |    (macro j_textarea::int (int int int) "j_textarea")                           |
137 |    (macro j_menubar::int (int) "j_menubar")                                     |
138 |    (macro j_menu::int (int string) "j_menu")                                    |
139 |    (macro j_helpmenu::int (int string) "j_helpmenu")                            |
140 |    (macro j_menuitem::int (int string) "j_menuitem")                            |
141 |    (macro j_checkmenuitem::int (int string) "j_checkmenuitem")                  |
142 |    (macro j_pack::void (int) "j_pack")                                          |
143 |    (macro j_print::void (int) "j_print")                                        |
144 |    (macro j_playsoundfile::void (string) "j_playsoundfile")                     |
145 |    (macro j_play::void (int) "j_play")                                          |
146 |    (macro j_sound::int (string) "j_sound")                                      |
147 |    (macro j_setfont::void (int int int int) "j_setfont")                        |
148 |    (macro j_setfontname::void (int int) "j_setfontname")                        |
149 |    (macro j_setfontsize::void (int int) "j_setfontsize")                        |
150 |    (macro j_setfontstyle::void (int int) "j_setfontstyle")                      |
151 |    (macro j_seperator::void (int) "j_seperator") ;; bug?                        |
152 |    (macro j_disable::void (int) "j_disable")                                    |
153 |    (macro j_enable::void (int) "j_enable")                                      |
154 |    (macro j_getstate::int (int) "j_getstate")                                   |
155 |    (macro j_getrows::int (int) "j_getrows")                                     |
156 |    (macro j_getcolumns::int (int) "j_getcolumns")                               |
```

```
157 |     (macro j_getselect::int (int) "j_getselect")                          |
158 |     (macro j_isselect::int (int int) "j_isselect")                        |
159 |     (macro j_isvisible::int (int) "j_isvisible")                          |
160 |     (macro j_isparent::int (int int) "j_isparent")                        |
161 |     (macro j_isresizable::int (int) "j_isresizable")                      |
162 |     (macro j_select::void (int int) "j_select")                           |
163 |     (macro j_deselect::void (int int) "j_deselect")                       |
164 |     (macro j_multiplemode::void (int int) "j_multiplemode")               |
165 |     (macro j_insert::void (int int string) "j_insert")                    |
166 |     (macro j_remove::void (int int) "j_remove")                           |
167 |     (macro j_removeitem::void (int string) "j_removeitem")                |
168 |     (macro j_removeall::void (int) "j_removeall")                         |
169 |     (macro j_setstate::void (int int) "j_setstate")                       |
170 |     (macro j_setrows::void (int int) "j_setrows")                         |
171 |     (macro j_setcolumns::void (int int) "j_setcolumns")                   |
172 |     (macro j_seticon::void (int int) "j_seticon")                         |
173 |     (macro j_setimage::void (int int) "j_setimage")                       |
174 |     (macro j_setvalue::void (int int) "j_setvalue")                       |
175 |     (macro j_setradiogroup::void (int int) "j_setradiogroup")             |
176 |     (macro j_setunitinc::void (int int) "j_setunitinc")                   |
177 |     (macro j_setblockinc::void (int int) "j_setblockinc")                 |
178 |     (macro j_setmin::void (int int) "j_setmin")                           |
179 |     (macro j_setmax::void (int int) "j_setmax")                           |
180 |     (macro j_setslidesize::void (int int) "j_setslidesize")               |
181 |     (macro j_setcursor::void (int int) "j_setcursor")                     |
182 |     (macro j_setresizable::void (int int) "j_setresizable")               |
183 |     (macro j_getlength::int (int) "j_getlength")                          |
184 |     (macro j_getvalue::int (int) "j_getvalue")                            |
185 |     (macro j_getscreenheight::int () "j_getscreenheight")                 |
186 |     (macro j_getscreenwidth::int () "j_getscreenwidth")                   |
187 |     (macro j_getheight::int (int) "j_getheight")                          |
188 |     (macro j_getwidth::int (int) "j_getwidth")                            |
189 |     (macro j_getinsets::int (int int) "j_getinsets")                      |
190 |     (macro j_getlayoutid::int (int) "j_getlayoutid")                      |
191 |     (macro j_getinheight::int (int) "j_getinheight")                      |
192 |     (macro j_getinwidth::int (int) "j_getinwidth")                        |
193 |     (macro j_gettext::string (int string) "j_gettext")                    |
194 |     (macro j_getitem::string (int int string) "j_getitem")                |
195 |     (macro j_getitemcount::int (int) "j_getitemcount")                    |
196 |     (macro j_delete::void (int int int) "j_delete")                       |
197 |     (macro j_replacetext::void (int string int int) "j_replacetext")      |
198 |     (macro j_appendtext::void (int string) "j_appendtext")                |
199 |     (macro j_inserttext::void (int string int) "j_inserttext")            |
200 |     (macro j_settext::void (int string) "j_settext")                      |
201 |     (macro j_selectall::void (int) "j_selectall")                         |
202 |     (macro j_selecttext::void (int int int) "j_selecttext")               |
203 |     (macro j_getselstart::int (int) "j_getselstart")                      |
204 |     (macro j_getselend::int (int) "j_getselend")                          |
205 |     (macro j_getseltext::string (int string) "j_getseltext")              |
206 |     (macro j_getcurpos::int (int) "j_getcurpos")                          |
207 |     (macro j_setcurpos::void (int int) "j_setcurpos")                     |
208 |     (macro j_setechochar::void (int char) "j_setechochar")                |
209 |     (macro j_seteditable::void (int int) "j_seteditable")                 |
210 |     (macro j_setshortcut::void (int char) "j_setshortcut")                |
211 |     (macro j_quit::void () "j_quit")                                      |
212 |     (macro j_kill::void () "j_kill")                                      |
213 |     (macro j_setsize::void (int int int) "j_setsize")                     |
214 |     (macro j_getaction::int () "j_getaction")                             |
215 |     (macro j_nextaction::int () "j_nextaction")                           |
216 |     (macro j_show::void (int) "j_show")                                   |
```

```
217 |     (macro j_showpopup::void (int int int) "j_showpopup")                |
218 |     (macro j_add::void (int int) "j_add")                                |
219 |     (macro j_release::void (int) "j_release")                            |
220 |     (macro j_releaseall::void (int) "j_releaseall")                      |
221 |     (macro j_hide::void (int) "j_hide")                                  |
222 |     (macro j_dispose::void (int) "j_dispose")                            |
223 |     (macro j_setpos::void (int int int) "j_setpos")                      |
224 |     (macro j_getviewportheight::int (int) "j_getviewportheight")         |
225 |     (macro j_getviewportwidth::int (int) "j_getviewportwidth")           |
226 |     (macro j_getxpos::int (int) "j_getxpos")                             |
227 |     (macro j_getypos::int (int) "j_getypos")                             |
228 |     (macro j_getpos::void (int int* int*) "j_getpos")                    |
229 |     (macro j_getparentid::int (int) "j_getparentid")                     |
230 |     (macro j_setfocus::void (int) "j_setfocus")                          |
231 |     (macro j_hasfocus::int (int) "j_hasfocus")                           |
232 |     (macro j_getstringwidth::int (int string) "j_getstringwidth")        |
233 |     (macro j_getfontheight::int (int) "j_getfontheight")                 |
234 |     (macro j_getfontascent::int (int) "j_getfontascent")                 |
235 |     (macro j_keylistener::int (int) "j_keylistener")                     |
236 |     (macro j_getkeycode::int (int) "j_getkeycode")                       |
237 |     (macro j_getkeychar::int (int) "j_getkeychar")                       |
238 |     (macro j_mouselistener::int (int int) "j_mouselistener")             |
239 |     (macro j_getmousex::int (int) "j_getmousex")                         |
240 |     (macro j_getmousey::int (int) "j_getmousey")                         |
241 |     (macro j_getmousepos::void (int int* int*) "j_getmousepos")          |
242 |     (macro j_getmousebutton::int (int) "j_getmousebutton")               |
243 |     (macro j_focuslistener::int (int) "j_focuslistener")                 |
244 |     (macro j_componentlistener::int (int int) "j_componentlistener")     |
245 |     (macro j_windowlistener::int (int int) "j_windowlistener")           |
246 |     (macro j_setflowlayout::void (int int) "j_setflowlayout")            |
247 |     (macro j_setborderlayout::void (int) "j_setborderlayout")            |
248 |     (macro j_setgridlayout::void (int int int) "j_setgridlayout")        |
249 |     (macro j_setfixlayout::void (int) "j_setfixlayout")                  |
250 |     (macro j_setnolayout::void (int) "j_setnolayout")                    |
251 |     (macro j_setborderpos::void (int int) "j_setborderpos")              |
252 |     (macro j_sethgap::void (int int) "j_sethgap")                        |
253 |     (macro j_setvgap::void (int int) "j_setvgap")                        |
254 |     (macro j_setinsets::void (int int int int int) "j_setinsets")        |
255 |     (macro j_setalign::void (int int) "j_setalign")                      |
256 |     (macro j_setflowfill::void (int int) "j_setflowfill")                |
257 |     (macro j_translate::void (int int int) "j_translate")                |
258 |     (macro j_cliprect::void (int int int int int) "j_cliprect")          |
259 |     (macro j_drawrect::void (int int int int int) "j_drawrect")          |
260 |     (macro j_fillrect::void (int int int int int) "j_fillrect")          |
261 |     (macro j_drawroundrect::void (int int int int int int int)           |
262 |               "j_drawroundrect")                                         |
263 |     (macro j_fillroundrect::void (int int int int int int int)           |
264 |               "j_fillroundrect")                                         |
265 |     (macro j_drawoval::void (int int int int int) "j_drawoval")          |
266 |     (macro j_filloval::void (int int int int int) "j_filloval")          |
267 |     (macro j_drawcircle::void (int int int int) "j_drawcircle")          |
268 |     (macro j_fillcircle::void (int int int int) "j_fillcircle")          |
269 |     (macro j_drawarc::void (int int int int int int int) "j_drawarc")    |
270 |     (macro j_fillarc::void (int int int int int int int) "j_fillarc")    |
271 |     (macro j_drawline::void (int int int int int) "j_drawline")          |
272 |     (macro j_drawpolyline::void (int int int* int*) "j_drawpolyline")    |
273 |     (macro j_drawpolygon::void (int int int* int*) "j_drawpolygon")      |
274 |     (macro j_fillpolygon::void (int int int* int*) "j_fillpolygon")      |
275 |     (macro j_drawpixel::void (int int int) "j_drawpixel")                |
276 |     (macro j_drawstring::void (int int int string) "j_drawstring")       |
```

```
277 |     (macro j_setxor::void (int int) "j_setxor")                          |
278 |     (macro j_getimage::int (int) "j_getimage")                           |
279 |     (macro j_getimagesource::void (int int int int int                   |
280 |            int* int* int*) "j_getimagesource")                           |
281 |     (macro j_drawimagesource::void (int int int int int                  |
282 |            int* int* int*) "j_drawimagesource")                          |
283 |     (macro j_getscaledimage::int (int int int int int int int)           |
284 |              "j_getscaledimage")                                         |
285 |     (macro j_drawimage::void (int int int int) "j_drawimage")            |
286 |     (macro j_drawscaledimage::void (int int int int int                  |
287 |              int int int int int) "j_drawscaledimage")                   |
288 |     (macro j_setcolor::void (int int int int) "j_setcolor")              |
289 |     (macro j_setcolorbg::void (int int int int) "j_setcolorbg")          |
290 |     (macro j_setnamedcolor::void (int int) "j_setnamedcolor")            |
291 |     (macro j_setnamedcolorbg::void (int int) "j_setnamedcolorbg")        |
292 |     (macro j_loadimage::int (string) "j_loadimage")                      |
293 |     (macro j_saveimage::int (int string int) "j_saveimage")              |
294 |     (macro j_sync::void () "j_sync")                                     |
295 |     (macro j_beep::void () "j_beep")                                     |
296 |     (macro j_random::int () "j_random")                                  |
297 |     (macro j_sleep::void (int) "j_sleep")                                |
298 |     ))                                                                   |
```