

Pilo's Visualization Tools for Scheme

By

David Pilo Mansion and Dr. Douglas Child

Department of Mathematics

And Computer Science

Contents:

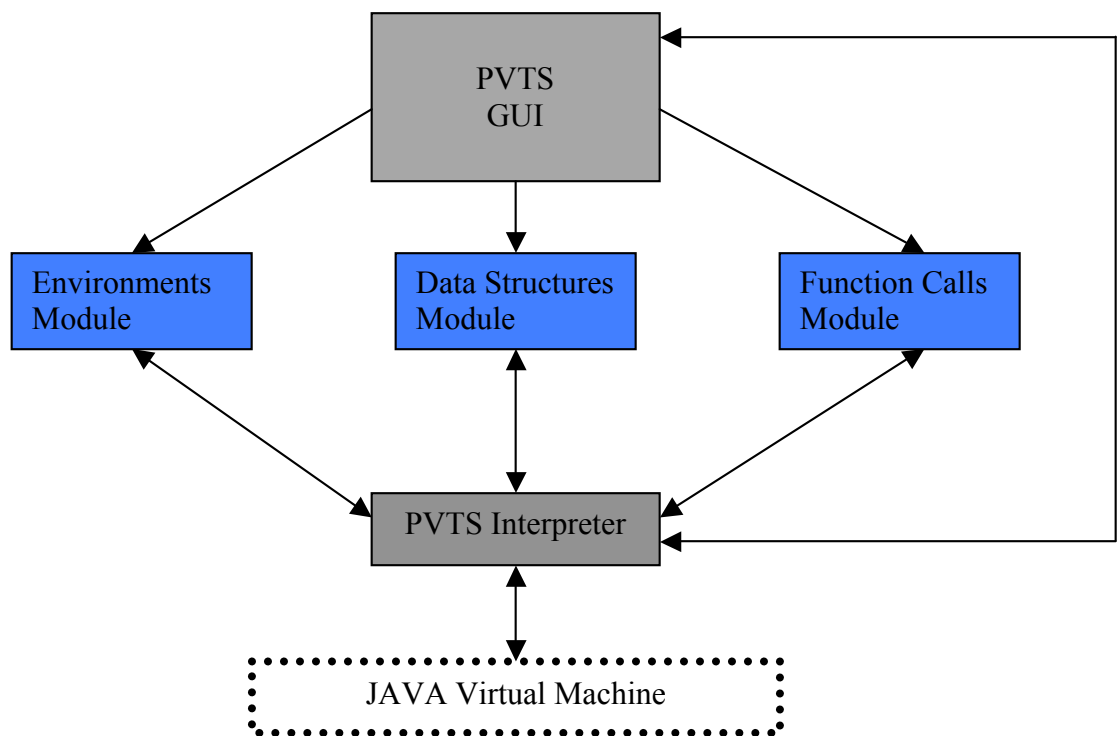
<i>Introduction</i>	3
The PVTS Interpreter.....	4
PVTS Main Window GUI Overview.....	4
Environment Viewer GUI Overview.....	5
Cons-cells Viewer GUI Overview.....	6
Function Calls Viewer GUI Overview.....	6
<i>Variables and Data Structures</i>	8
Scheme Data Structure Overview.....	8
Issues Representing Data Creation and Manipulation.....	8
The Algorithm.....	9
Other Issues.....	10
<i>Function Calls</i>	11
Scheme Function Calls Overview.....	11
Issues Representing Data Creation and Manipulation.....	12
The problem and the algorithm for its solution.....	13
<i>The Global Environment</i>	15
Scheme Global Environment Overview.....	15
<i>Acknowledgements - References</i>	16
<i>Appendix A – Interpreter Grammar</i>	17
<i>Appendix B - Examples</i>	20
The lists example.....	20
The append example.....	20
The factorial example.....	21
The Fibonacci example.....	22
The bank account example.....	22
The sorting algorithms example.....	23
<i>Appendix C – Code</i>	25
Cons-cells display algorithm.....	25
Function-calls trees display algorithm.....	32
<i>Appendix D – Work Log</i>	38
5/15/06:.....	38
6/18/06:.....	54
9/5/06:.....	64

Introduction

Students who know procedural and object-oriented languages frequently have difficulty learning the functional paradigm. The purpose of this work is to facilitate this transition by designing and implementing a set of visual tools that help students understand how Scheme, a functional language, programs work. To achieve our goals we worked on the implementation of a Scheme interpreter and a set of visual tools for different key aspects of functional programming languages. Pilo's Visualization Tools for Scheme (PVTS) emphasizes on the functional programming language paradigm and its visual representations. PVTS can be used by teachers as a teaching tool as well as by students as a learning tool.

This research is an attempt to further develop academic tools for the functional programming language Scheme. However because Scheme is a dialect of LISP, this software can be of great value for students of LISP. The software implemented consists in a fairly complex interpreter of Scheme and a set of three visual tools. These three visual tools or aids represent three main aspects of the functional programming language: data structures and lists, function calls, and the global environment.

This research is based on our own version of a limited Scheme interpreter written in Java 1.5 (See Appendix A for the complete grammar). Because the visual tools display the information that is being processed by the interpreter, they depend on capabilities of the interpreter. We needed an interpreter as functional as possible in order to run complex and varied samples of Scheme programs.



The Scheme Interpreter

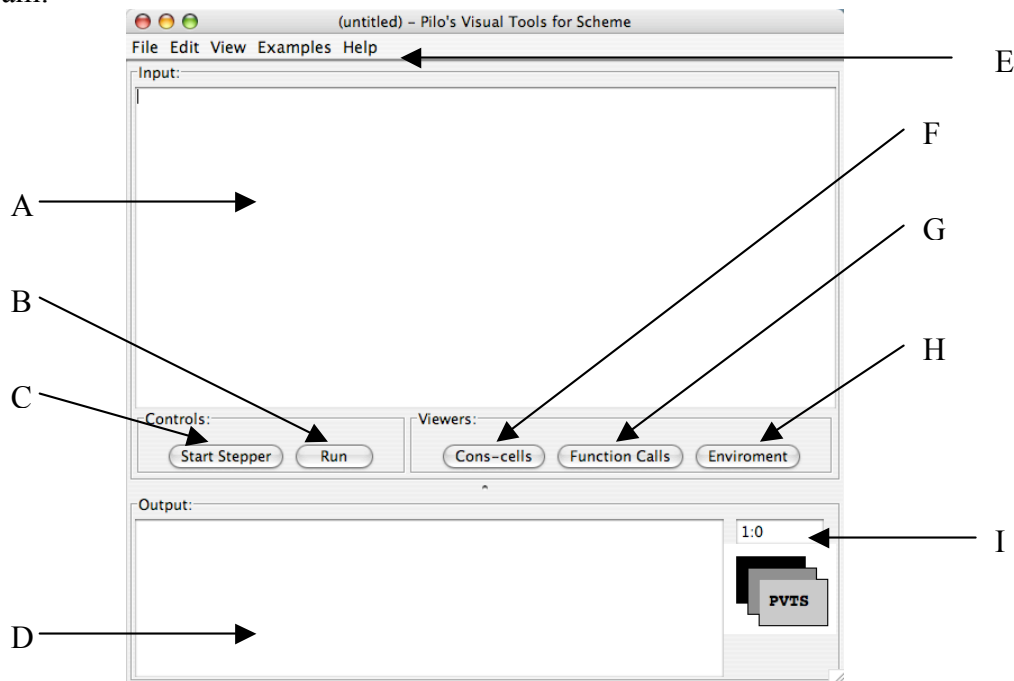
The PVTS Interpreter

As mentioned in the introduction, the tools provided by this software depend on a solid and complex Scheme interpreter. This interpreter was initially based on a CLI (Command Line Interface) that could process a small subset of the Scheme language with a single command per input. Now the interpreter and the three tools have a powerful GUI (Graphic User Interface) that supports multiple commands at a time as well as file I/O support.

This interpreter has been written using Java 5 as well as the rest of the PVTS software. Even though java will impose certain restrictions on the performance of the interpreter, PVTS will be able to run on any platform that supports the Java Virtual Machine 5 and above. For learning and teaching purpose I have found that the performance of the interpreter is quite satisfying. The main restriction though on this interpreter is the grammar of the interpreter. Far from supporting the R5RS¹ standard for Scheme, my interpreter has an extensive grammar of about 100 key words. I have based my grammar on a subset of the advanced language grammar suggested by DrScheme². A complete description of the PVTS grammar is included in this paper (See Appendix A). In order to ensure the proper functionality of the interpreter I used DrScheme to compare the outputs and learn the insights of the Scheme language.

PVTS Main Window GUI Overview

The GUI is powerful but very easy to use. Here is how it looks like once you launch the program:



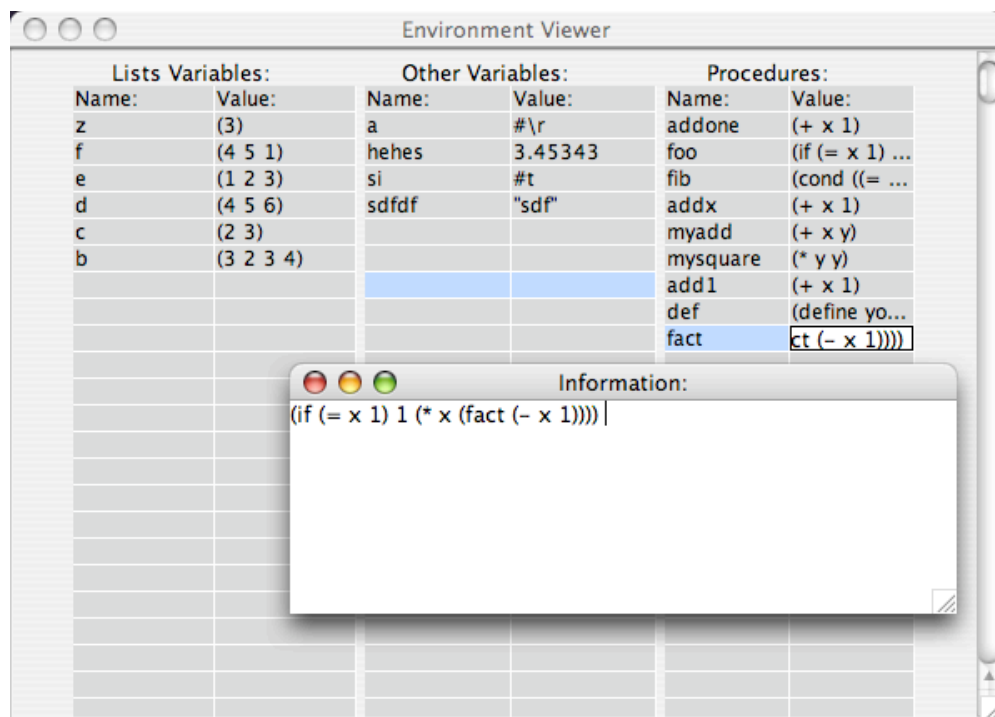
¹ <http://www.schemers.org/Documents/Standards/R5RS/>

² <http://www.plt-scheme.org/software/drscheme/>

The text editor (A) allows the user can input the code he would like to run the interpreter on. The user can select from the menu (E) to open an existing source file, or save a source file – the user will also be able to print files or access the preferences panel. The output of the interpreter (D) is where all the text-based results from running the interpreter will be printed as well as any syntactic errors. Controls (F), (G), and (H) will toggle on/off the three visualization modules. In order to execute a code the user will have to click on “run” (B), the user can also invoke the stepper mode by clicking on “Start Stepper” (C), which is a step-by-step interpretation of the code. A small display (I) indicates the position of the cursor in the text editor in the format <line : column>.

Environment Viewer GUI Overview

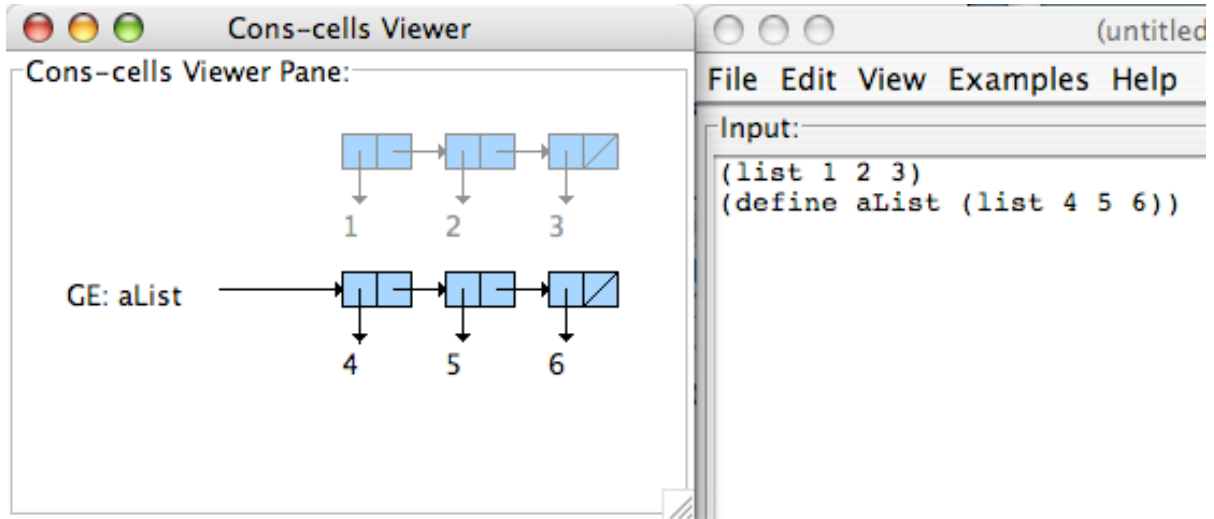
The visual module for the Environments represented in the following figure is simple and to the point:



This window displays information on the global environment that is currently in the memory of the interpreter. It displays by default three tables, the “list variables” table will display all the lists that have been bounded to a variable name. The “other variables” table displays all the variables that are bounded to a variable name that are not lists or procedures, i.e., a Boolean will be shown in this table. The “procedures” table displays the user-defined procedures that are present in the global environment. In addition, an information window pops-up containing the information of a cell clicked by the user.

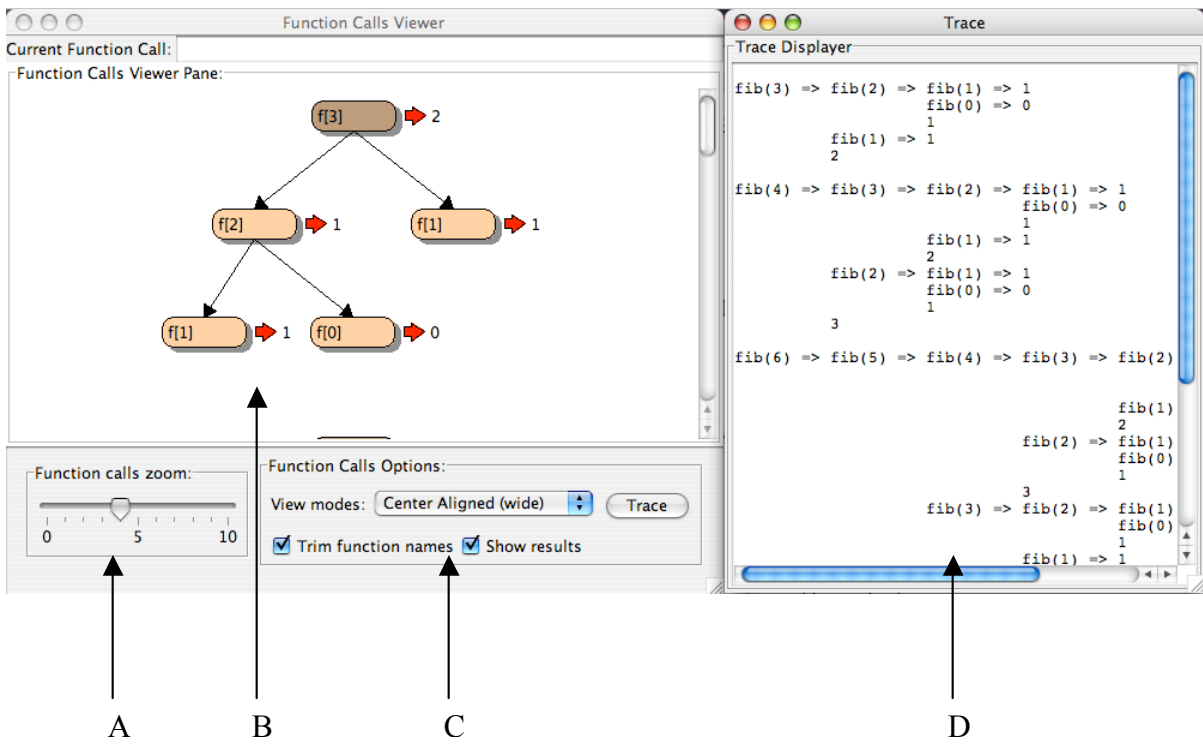
Cons-cells Viewer GUI Overview

This view very simple, it has no controls and displays the list that are being processed by the interpreter:



Function Calls Viewer GUI Overview

This is the most interactive module of the three. The window looks like this:



The main window (B) contains all the functions calls are represented visually. The zoom control (A) increases or decreases the size of the visual representations. In addition, an options pane (C) is available where different controls are offered in order to have the optimum visualization for a specific case. The options pane contains controls for different view modes that change the display of the ovals in the panel – a total of four view modes are available depending on alignment and space between ovals. Also, a control for trimming the names of the functions is available as well as a control to toggle on/off the visualization of each call's result. In addition, the user can see the trace of the function calls in a separate window (D).

Variables and Data Structures

Scheme Data Structure Overview

Scheme is a type-less language, which is based on two main data structures: atoms and lists. Atoms are a single type of data such as integers, strings or booleans, while lists are a collection of atoms and or lists. Scheme uses a linked list type of implementation. A linked list is composed of cons-cells or a single memory block pointing to *null* in the case of an empty list. A cons-cell is a pair of memory cells each pointing either to a list, an atom or *null*. In order to visually represent lists we used a widely accepted iconography, i.e. the representation of the list containing the atoms: the integer 1, the string “hello” and the float 2.4, is illustrated in the following example.

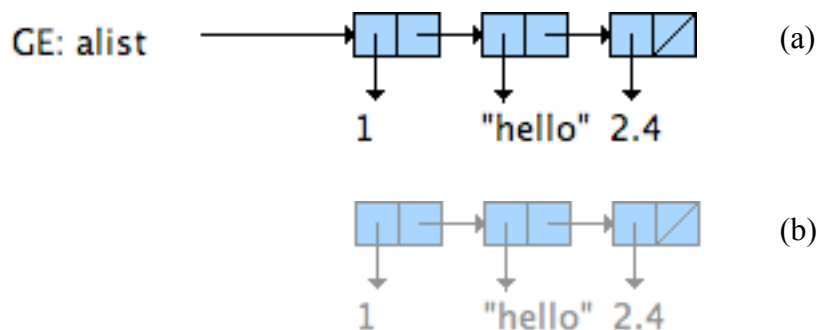


Figure 1. Figure (a) represents a list that is bound to a variable, that is, the list is accessible by the programmer through an address in memory. Figure (b) is a non-bound list, therefore not accessible through a variable.

In Scheme, lists can be created, bound, accessed and modified by the programmer. The manipulation of complex data structures involving lists can be hard to follow without a visual representation. Our project creates a visual interpretation of the data as our interpreter is processing it.

Issues Representing Data Creation and Manipulation

A common way to allocate space in memory for a list is by executing the command `(list <elements>)` or `'(<elements>)` where the elements are any number of atoms and/or lists. For instance the command `(list 1 "hello" 2.4)` will create the list seen in Figure 1. In order to display simple lists (lists containing only atoms) it was sufficient to add cons-cells expanding horizontally. In the case of lists containing lists the previous method was not sufficient due to possible graphical collisions. A graphical collision happens when a cons-cell is sharing its position with another cons-cell or atom as illustrated below.

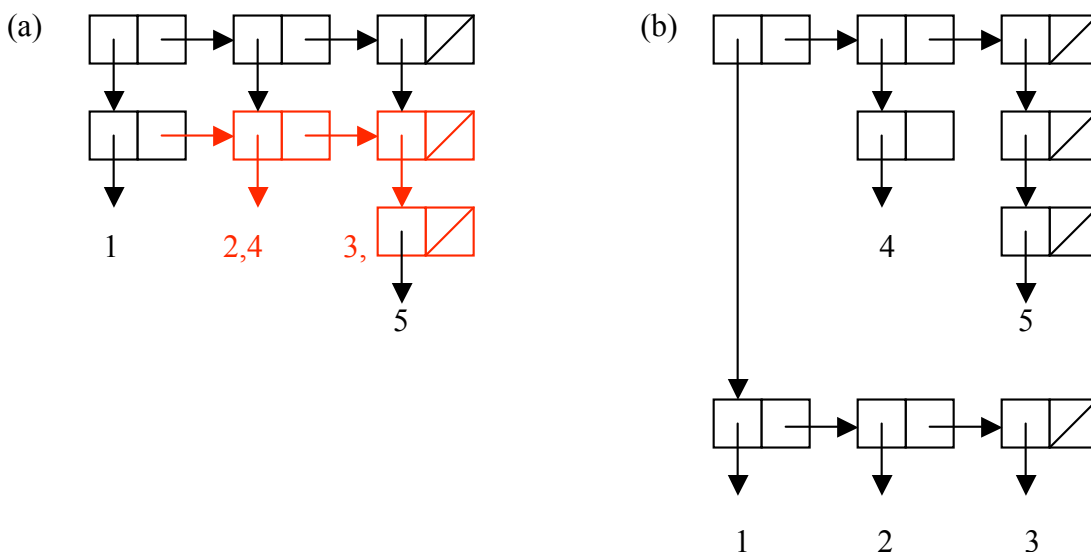


Figure 2. The list is used in this example is the list: '((1 2 3) (4) ((5)))'. In Figure (a) the cons-cells and atoms in red represent collisions. Figure (b) is the same list without collisions.

The Algorithm

The algorithm implemented in order to solve this problem consists in creating an initial matrix used as a “grid” for the display of the list and perform operations on it until there are no collisions. Note that in order to exist a collision in the matrix, there has to be at least an element of the matrix a_{ij} where there is two or more elements. The elements in the matrix will be numbers. These numbers correspond to the labels given to the sub-lists. For example, the list in Figure 2 (a) and the label of its sub-lists is exemplified below.

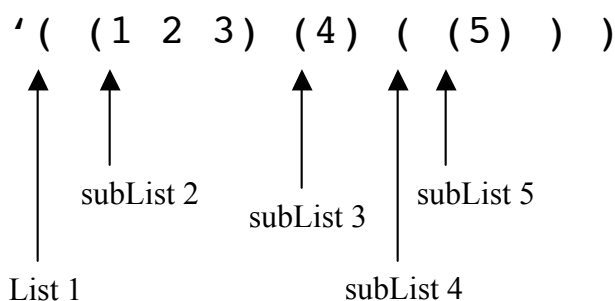


Figure 3. Represents the list used in Figure 2 (a) and (b) with its labels.

The algorithm will store in a matrix the lists. If an atom or a cons-cell belonging to list x is in the a_{ij} spot then the reference to that list will be stored in the a_{ij} element of the matrix. If there is a collision between an atom and a cons-cell, the list to which the cons-cell belongs to will be shifted down one spot, if the collision happens between two atoms or two cons-cells then the oldest list (the furthest left) will be shifted down. This process is repeated until there are no collisions.

(a)

1	1	1
2	2,3	2,4
2	2,3	2,5
		5

(b)

1	1	1
	3	4
	3	5
		5
2	2	2
2	2	2

Figure 4. Figure (a) represents the initial matrix with collisions. Figure (b) represents the matrix after the algorithm has been performed, thus without collisions.

Note the visual representations of the list illustrated in Figure 2 correspond with the matrices in Figure 4. Figures 2 (a) and 4(a) represent the list before the application of the algorithm; Figures 2 (b) and 4(b) correspond to the final representation of the list and its matrix without collisions.

The code for this algorithm can be found in Appendix C.

Other Issues

Another issue concerning the display of lists was to be able to represent in a transparent manner how Scheme handles list mutations. When a mutation occurs, the data in the list is changed. Because of these changes, sub-lists might be part of data that is no longer accessible by the program and therefore are placed in the pool of data that is going to be reallocated by the garbage collector. In an effort to preserve as much useful information to the user we decided to keep the data that is no longer accessible and just grey it out as shown below.

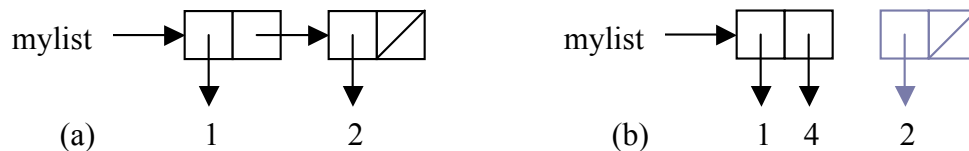


Figure 5. Figure (a) is the list containing the integers 1 and 2. Figure (b) represents a mutation of the previous list where the second cons-cell is no longer accessible. Note Figure (b) is a dotted pair, that is, a cons-cell where both memory cells point to an atom.

Function Calls

Scheme Function Calls Overview

Scheme, like other functional programming languages, makes extensive use of recursion. Recursive functions call themselves (See Figure 6a for a recursive version of a Fibonacci function in Scheme). Fibonacci and factorial algorithms are widely used for their easy and elegant recursive definition (See Appendix B for factorial definition).

```
fib(n) = fib(n-1) + fib(n-2)
fib(1) = 1
fib(0) = 0
```

Figure 6a. This Scheme code is a recursive definition for the Fibonacci function.

The visualization of function calls is a static flow diagram where function instances are represented as ovals, and function calls are represented as arrows from the caller function to the called function (See Figure 6b). The user can see the full information of a given call by clicking on a given oval; the information will be displayed in the top of the window.

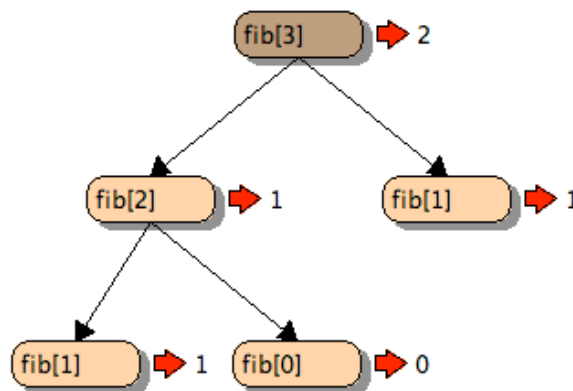


Figure 6b. This diagram represents the Fibonacci function when passed the integer 3. Black arrows represent a function call, red arrows point to the result from the evaluation of a given function.

Pilo’s Visualization Tools for Scheme also includes a trace visualization of the function calls. The visual output (See Figure 7) is clean and simple since some functions can have a high level of complexity.

```
fib(3) => fib(2) => fib(1) => 1
                               fib(0) => 0
                               1
                                fib(1) => 1
                                2
```

Figure 7. This is the trace of the evaluation of the Fibonacci function when passed the integer 3. Indentation is used to keep track of which function calls which function.

Issues Representing Data Creation and Manipulation

The hardest part of displaying function calls was to choose the placement for each oval. For simply recursive functions like factorial, the allocation of ovals in the panel is rather straightforward – the ovals are placed following a vertical direction. The difficulties increased when displaying multiply recursive functions such as quickSort.

Displaying the evaluation of Fibonacci wasn't too much of a problem since this algorithm grows more or less equally from both sides, that is Fibonacci will always be somewhat balanced (See Figure 8).

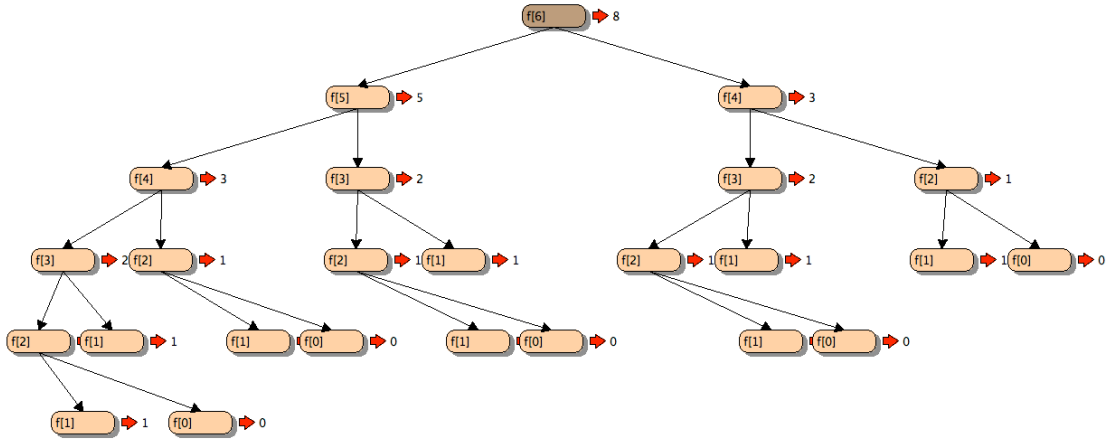


Figure 8. This is the tree of the evaluation of the Fibonacci function when passed the integer 6. This figure illustrates the balance of Fibonacci.

However, some functions do not grow symmetrically. Some of them they expand in a very unbalanced way, that is, the tree grows a lot from one side and little from the other. The algorithm for insertion sort is a good example (See Figure 9).

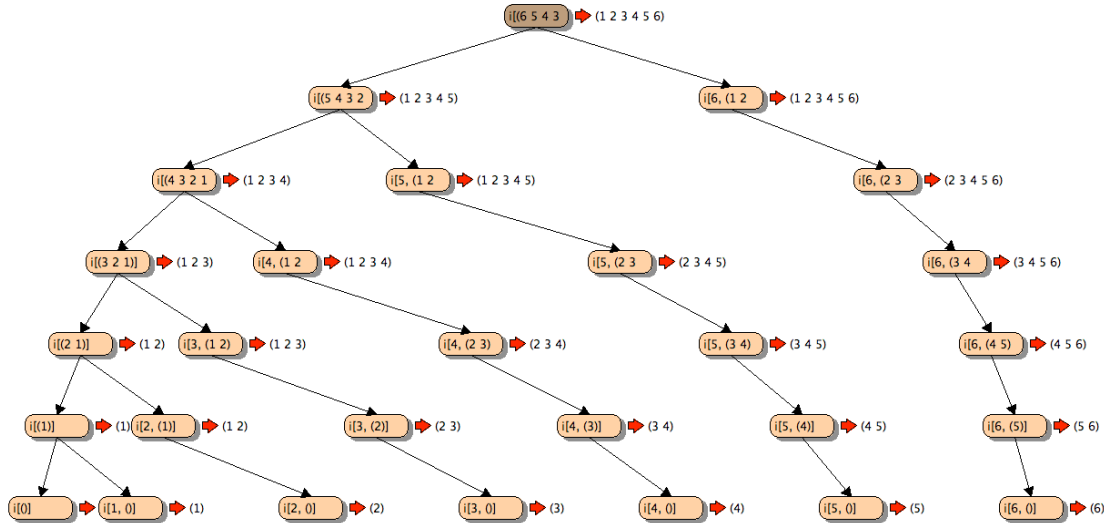


Figure 9. This is the tree of the evaluation of the insertionSort function when passed the list (6 5 4 3 2 1). This figure illustrates the unbalance of insertionSort.

The algorithm that I developed in order to visualize function calls takes into account how the tree grows. As the tree grows it draws the ovals in such a way that the drawing space is efficiently used. In order to achieve this, the algorithm takes into account the number of calls that have been made on a certain level and displays the ovals (or the calls) in a balance or symmetrical way. Figure 10 illustrates the capacity of this displaying algorithm to visualize, in an elegant way, complex function calls.

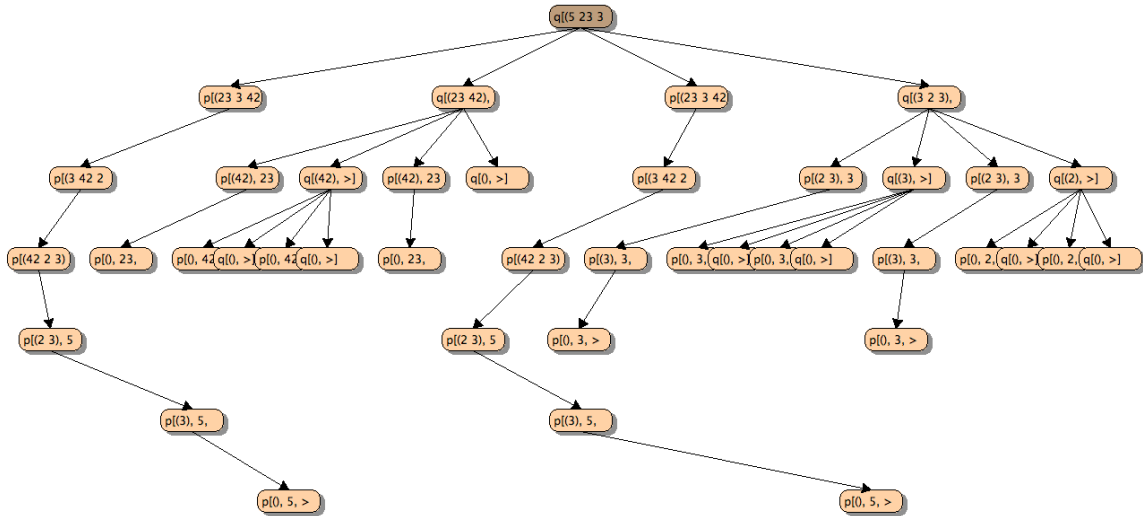


Figure 10. This is the tree of the evaluation of the quickSort function when passed the list (5 23 3 42 2 3). This figure illustrates the complexity of functions such as quickSort. This function makes two calls on another function called pivotlist and two calls on quickSort itself. See Appendix B for the code.

The problem and the algorithm for its solution

The problem was to be able to represent a tree of function calls such that any multiply recursive call can be displayed. I assumed that having all the nodes that have same depth are positioned in the same horizontal line in the panel. So the problem was to figure out what was going to be the horizontal position for each node. The aesthetic rules that I adopted are described in the following:

1. A parent node should be as centered as possible relative to its descendents.
2. Two different nodes cannot occupy the exact same space.
3. The width of the tree should not be bigger than the panel (area of display) width.

The algorithm I have implemented uses a matrix that stores the tree information. The first function call is made and its instance is placed in the first row first column of the matrix; the current size of the Stack of environments is k . When a function calls another function (or itself) an environment is pushed into the Environment Stack and that instance of function call is added into the matrix in the last non-used cell of the $(i - k)^{th}$ row, where i is the size of the Stack of Environments. Before an environment is popped a “dummy” node is inserted in

the last non-used cell of the $(i - k + 1)^{th}$ row. If the current function call instance is the root then only a dummy node is inserted in the first row. For example the matrix for the Fibonacci function in Figure 6 is show in Figure 11.

Row 1				
Row k+1	Fib(3)			
Row k+2	Fib(2)	Fib(1)	Dummy	
Row k+3	Fib(1)	Fib(0)	Dummy	Dummy
Row k+4	Dummy	Dummy		

Figure 11. Represents the matrix created for the tree illustrated in Figure 6b when the last recursive call is made and $i = k+4$.

After having generated the matrix we can draw the tree. Each function call instance of a row will be drawn in the same horizontal line and instances from two different rows will be on the same horizontal line. Dummy nodes won't be drawn. Assume that the width of the drawing panel is 100 units wide. A node a_{ij} will be drawn at $\frac{100}{n} \times j$ units in the $x - coordinate$, where n is the number of elements in the i^{th} row.

Given an element in the ij^{th} position, its parent will be somewhere in the $(i - 1)^{th}$ row. The parent will be the $(k + 1)^{th}$ non-dummy node in the $(i - 1)^{th}$ row, where k is the number of dummy nodes in the i^{th} row. This is the information that the process that draws the arrows will use.

I have found that this algorithm was really good for fairly complex trees. For instance looking at the tree in Figure 10, the two nodes at the bottom are equally sharing the width of the drawing area. Because of that, if the tree would grow from these nodes they would have been in an ideal position to grow further.

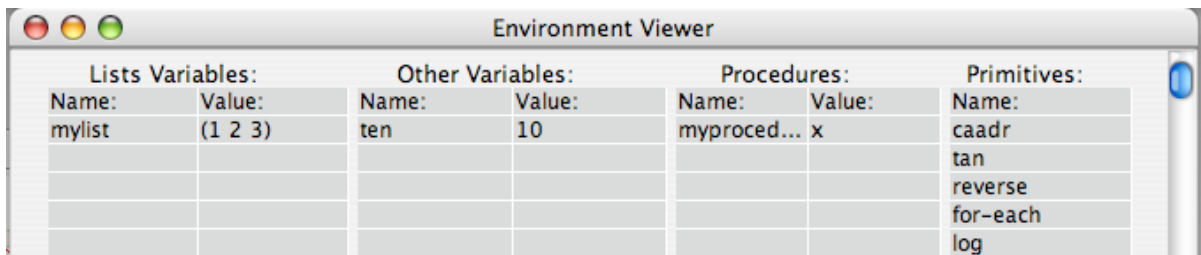
The code for this algorithm can be found in Appendix C.

The Global Environment

Scheme Global Environment Overview

Scheme, as most functional programming languages, uses nested environments. The main environment or global environment is the first one that is created when the interpreter first starts its execution. This global environment holds the primary operators such as <primitive: +> or <primitive: list>. This environment also might be used to save user created variables or procedures.

In order to represent this data, I decided to have it shown in different tables depending on the category: procedures, lists, primitives, and other variables. The four tables can be displayed simultaneously indicating everything that is stored in the global environment (See Figure 12)



The screenshot shows a window titled "Environment Viewer" with four columns of data:

Lists Variables:		Other Variables:		Procedures:		Primitives:
Name:	Value:	Name:	Value:	Name:	Value:	Name:
mylist	(1 2 3)	ten	10	myproced...	x	caadr
						tan
						reverse
						for-each
						log

Figure 10. This represents the global environment.

Acknowledgements - References

1. API specification for the Java 2 Platform, Standard Edition, version 1.4.2
Sun Microsystems. <<http://java.sun.com/j2se/1.4.2/docs/api/index.html>>
2. Structure and Interpretation of Computer Programs – 2nd Edition
Harold Abelson and Gerald Jay Sussman. <<http://mitpress.mit.edu/sicp/>>
3. A Survey of Program Visualizations for the Functional Paradigm, Jaime Urquiza-Fuentes, J. Angel Velazquez-Iturbide <www.dcs.warwick.ac.uk/pvw04/p01.pdf>

Appendix A – Interpreter Grammar

Interpreter Language:

- **PROGRAM**
 - DEF-OR-EXP ...
- **DEF-OR-EXP**
 - **DEFINITION**
 - **EXPRESSION**
- **DEFINITION**
 - (define (NAME NAME ...) EXPRESSION)
 - (define NAME EXPRESSION)
- **EXPRESSION**
 - (set! NAME EXPRESSION)
 - (lambda (NAME ...) EXPRESSION)
 - (let ((NAME EXPRESSION) ...) EXPRESSION)
 - (PRIM-OP EXPRESSION ...)
 - (cond (EXPRESSION EXPRESSION) (EXPRESSION EXPRESSION) ...)
 - (cond (EXPRESSION EXPRESSION) ... (else EXPRESSION))
 - (if EXPRESSION EXPRESSION EXPRESSION)
 - (and EXPRESSION EXPRESSION EXPRESSION ...)
 - (or EXPRESSION EXPRESSION EXPRESSION ...)
 - **NAME**
 - **'QUOTED**
 - **NUMBER**
 - **STRING**
 - **CHARACTER**
- **NAME**
 - a sequence of keyboard characters not including: space " , ' ` () [] { } | ; #
- **QUOTED**
 - **NAME**
 - **NUMBER**
 - **STRING**
 - **CHARACTER**
 - **'QUOTED**
- **PRIM-OPs**
 - *Numbers: Integers, Reals.*
 - *: (num num num ... -> num)
 - +: (num num num ... -> num)
 - -: (num num ... -> num)
 - /: (num num num ... -> num)
 - <: (real real real ... -> boolean)
 - <=: (real real real ... -> boolean)
 - =: (num num num ... -> boolean)
 - >: (real real real ... -> boolean)
 - >=: (real real ... -> boolean)
 - abs: (real -> real)

- `acos` : (num -> num)
- `asin` : (num -> num)
- `atan` : (num -> num)
- `ceiling` : (real -> int)
- `cos` : (num -> num)
- `even?` : (integer -> bool)
- `exp` : (num -> num)
- `expt` : (num num -> num)
- `floor` : (real -> int)
- `gcd` : (int int -> int)
- `integer?` : (any -> bool)
- `log` : (num -> num)
- `max` : (real real ... -> real)
- `min` : (real real ... -> real)
- `modulo` : (int int -> int)
- `negative?` : (number -> bool)
- `number?` : (any -> boolean)
- `odd?` : (integer -> bool)
- `positive?` : (number -> bool)
- `real?` : (any -> bool)
- `sin` : (num -> num)
- `sqrt` : (num -> num)
- `tan` : (num -> num)
- `zero?` : (number -> bool)
- *Booleans*
 - `boolean?` : (any -> boolean)
 - `not` : (boolean -> boolean)
- *Symbols*
 - `symbol?` : (any -> boolean)
- *Lists*
 - `append` : ((listof any) ... -> (listof any))
 - `car` : ((cons Y (listof X)) -> Y)
 - `cdr` : ((cons Y (listof X)) -> (listof X))
 - `caar` : ((listof any)) -> any
 - `cadr` : ((listof any) -> (listof any))
 - `caaar` : ((listof any)) -> any
 - `caadr` : ((listof any) -> (listof any))
 - `caaaaar` : ((listof any)) -> any
 - `caaaadr` : ((listof any) -> (listof any))
 - `cadar` : ((listof any)) -> any
 - `caddr` : ((listof any) -> (listof any))
 - `cadaar` : ((listof any)) -> any
 - `cadadr` : ((listof any) -> (listof any))
 - `caadar` : ((listof any)) -> any
 - `caaddr` : ((listof any) -> (listof any))
 - `cdar` : ((listof any)) -> any

- `cddr`: ((listof any) -> (listof any))
- `cdaar`: ((listof any)) -> any
- `cdadr`: ((listof any) -> (listof any))
- `cdaaar`: ((listof any)) -> any
- `cdaadr`: ((listof any) -> (listof any))
- `cddar`: ((listof any)) -> any
- `cdddr`: ((listof any) -> (listof any))
- `cdbaar`: ((listof any)) -> any
- `cddadr`: ((listof any) -> (listof any))
- `cdddar`: ((listof any)) -> any
- `cdddrdr`: ((listof any) -> (listof any))
- `caddar`: ((listof any)) -> any
- `cadadr`: ((listof any) -> (listof any))
- `cdadar`: ((listof any)) -> any
- `cons`: (X (listof X) -> (listof X))
- `length`: (list -> number)
- `list`: (any ... (listof any) -> (listof any))
- `list`: (any ... -> (listof any))
- `list?`: (any -> boolean)
- `member`: (any list -> (union false list))
- `null?`: (any -> boolean)
- `reverse`: (list -> list)
- `set-car!`: ((cons Y (listof X)) Y -> void)
- `set-cdr!`: ((cons Y (listof X)) (listof X) -> void)
- *Characters*
 - `char?`: (any -> boolean)
- *Strings*
 - `string?`: (any -> boolean)
- *Misc*
 - `eq?`: (any any -> boolean)
 - `equal?`: (any any -> boolean)
 - `eqv?`: (any any -> boolean)
- *Higher-Order Functions*
 - `apply`: ((X-1 ... X-N -> Y) X-1 ... X-i (list X-i+1 ... X-N) -> Y)
 - `for-each`: ((any ... -> any) (listof any) ... -> void)
 - `map`: ((X ... -> Z) (listof X) ... -> (listof Z))
 - `procedure?`: (any -> boolean)
- *Printing*
 - `display`: (any -> void)
 - `newline`: (-> void)

Appendix B - Examples

In this section I will discuss examples of interest that illustrate important features of Scheme. All these examples can be directly accessed from the program from the “Examples” menu in the main menu bar of the GUI.

The lists example

```
; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; NOTE: for this example use the Cons-cells Viewer.

"LISTS EXAMPLE:"
; There is different ways to create lists:
'(1 2 3)
(list 1 2 3)
(cons 1 (cons 2 (cons 3 ())))
; lists are heterogeneous:
'(1 1.4 #f "hello" #\j '(1 2 3))
; car accesses the first element of a list
(car '(1 2 3))
; while cdr accesses the rest of the list
(cdr '(1 2 3))
; There is also what we call dotted pairs:
(define a (cons 1 2))
(define b (cons 1 (cons 2 ())))
; what is the difference between a and b?
a b
```

The append example

```
; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; NOTE: for this example use the Cons-cells Viewer.
;
"APPEND EXAMPLE:"
; This example is a very interesting one since first time
; Scheme users don't know how the append function works. In
; plain English, append takes any number of lists and appends
; them together. That is:
; (append '(1 2 3) '(4 5 6)) evaluates to
; (append '(1 2 3) '(4 5 6))
; There are several ways that append could work with the cons-cells in the
; lists passed to it.
; It could make copies of one or more of the lists or; it could just
; connect the existing cells.; Let's see how it actually works.
(define a '(1 2 3))
(define b '(4 5 6))
; lets append both lists and call it c
(define c (append a b))
```

```

; c evals to
c
; Now lets modify the list a
(set-car! a 0)
; so a evals to
a
; and b evals to
b
; and c evals to
c
; c evaluates to (1 2 3 4 5 6) WHY?
; append copied the list a and appended that copy
; to the list b. So if we modify the list b
(set-car! b 0)
; b evals to
b
; c evals to
c
; -----
; Let's implement our own version of append
; where this time append will just connect the existing cells
(define (last l)
  (if (null? (cdr l))
      l
      (last (cdr l))))
(define (append! l1 l2)
  (set-cdr! (last l1) l2)l1)
(define list1 '(20 21 22))
(define list2 '(23 24 25))
(define list3 (append! list1 list2))
"END OF APPEND EXAMPLE"

```

The factorial example

```

; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; NOTE: for this example use the Function Calls Viewer.
;
"FACTORIAL EXAMPLE:"
; The factorial function is a widely use in
; functional programming languages to introduce
; recursive function calls. The mathematical notation
; for factorial is  $f(n) = n!$ 
; We usually define it as  $f(n) = n \times f(n-1)$ 
;  $f(1) = 1$ 
; Here is an example of how it works:
;  $f(3) = 3 \times f(2)$ 
;  $f(3) = 3 \times 2 \times f(1)$ 
;  $f(3) = 3 \times 2 \times 1$ 
;  $f(3) = 3 \times 2$ 
;  $f(3) = 6$ 
; Lets define factorial:
(define (factorial n)
  (if (= n 1)
      1

```

```

      (* n (factorial (- n 1))))))
; (factorial 3) evaluates to
(factorial 3)
; (factorial 4) evaluates to
(factorial 4)
; (factorial 12) evaluates to
(factorial 12)

```

The Fibonacci example

```

; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; NOTE: for this example use the Function Calls Viewer.
;
" FIBONACCI EXAMPLE:"
; The Fibonacci function is a widely use in
; functional programming languages to introduce
; doubly recursive function.
; We usually define it as fib(n) = fib(n-1) + fib(n-2)
;
;           fib(1) = 1
;           fib(0) = 0
; Here is an example of how it works:
; fib(3) = fib(2) + fib(1)
; fib(3) = fib(1) + fib(0) + fib(1)
; fib(3) = 1 + 0 + 1
; fib(3) = 2
; Lets define Fibonacci:
(define (fib n)
  (cond
    ((= n 1) 1)
    ((= n 0) 0)
    (else (+ (fib (- n 1)) (fib (- n 2))))))
; (fib 3) evaluates to
(fib 3)
; (fib 4) evaluates to
(fib 4)
; (fib 6) evaluates to
(fib 6)

```

The bank account example

```

; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; "BANK ACCOUNT EXAMPLE:"
; The bank account problem because it evidences
; the fact that, in scheme, procedures have their own
; environment.
(define (make-account balance)
  (define (deposit amount)
    (cond
      ((>= amount 0) (set! balance (+ balance amount))

```

balance)

```

                (else "Can't deposit a negative amount.)))
  (define (withdraw amount)
    (set! balance (- balance amount)) balance)
  (define (dispatch s)
    (cond
      ((eq? s 'withdraw) withdraw)
      (else deposit)))
  dispatch)
(define acc1 (make-account 500))
; what is acc1?
acc1
; what is (acc1 'withdraw)
(acc1 'withdraw)
; lets withdraw $70 from the account and then make a $30 deposit.
((acc1 'withdraw) 70)
((acc1 'deposit) 30)
; why is there $460 left in the account?
; acc1 has its own enviroment where balance is defined.

```

The sorting algorithms example

```

; By David A. Pilo Mansion
; davidpilo@gmail.com
; 2007
;
; NOTE: for this example use both the Cons-cells Viewer
;       and the Function Calls Viewer.

"SORTING ALGORITHMS EXAMPLE:"
;insertion sort:
(define (insertionSort lis)
  (cond
    ((null? lis) lis)
    (else (insertElmt (car lis) (insertionSort (cdr lis))))))

(define (insertElmt ele lis)
  (cond
    ((null? lis) (list ele))
    (< ele (car lis)) (cons ele lis))
    (else (cons (car lis) (insertElmt ele (cdr lis))))))

(insertionSort '(1 2 3 4))
(insertionSort '(4 3 2 1))

;qsort test:
(define (qsort lis order)
  (define (pivotlist lis pivot order switch)
    (cond
      ((null? lis) (list))
      ((= 1 switch) (if (order pivot (car lis)) (cons (car lis) (pivotlist
(cdr lis) pivot order 1)) (pivotlist (cdr lis) pivot order 1)))
      (else (if (not(order pivot (car lis))) (cons (car lis) (pivotlist
(cdr lis) pivot order 0)) (pivotlist (cdr lis) pivot order 0)))))

  (cond
    ((null? lis) (list))

```

```
(else (append(append (qsort (pivotlist (cdr lis) (car lis) order 0)
order) (list(car lis))) (qsort (pivotlist (cdr lis) (car lis) order 1)
order))))))
```

```
(qsort '(5 23 3 42 2 3) >)
```

Appendix C – Code

Cons-cells display algorithm

The following java class (ListManager.java) contains the algorithm used to display the cons-cells.

```

/*
 * ListManager.java
 *
 * author: David A. Pilo Mansion
 *
 *
 * Pilo's Visualization Tools for Scheme (PVTS). A Basic Visual Scheme Interpreter.
 * Copyright (C) 2007 David A. Pilo Mansion
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * Contact information:
 * David Pilo
 * davidpilo@gmail.com
 */

import java.util.*;

/* ListManager Class by David Pilo Mansion.
 *
 *
 * The algorithm will store in a matrix the lists. If an SEInt or a cons-cell
 * belonging to list x is in the
 * (i,j) spot then the reference to that list will be stored in the (i,j) element
 * of the matrix.
 * If there is a collision between an SEInt and a cons-cell, the list to which the
 * cons-cell belongs to will be shifted down one spot,
 * if the collision happens between two SEInts or two cons-cells then the oldest
 * list (the furthest left) will be shifted down.
 * The Algorithm will repeat the process until no collisions are found.
 *
 */

```

```

class ListManager {
    int xMax;
    int yMax;
    int xOffset;
    int yOffset;
    int xStep = 32;
    int yStep = 48;
    int MAX = 30;
    ListCC tList;
    SExpression current;
    SExpression cu;
    ListMatrix listMatrix;
    LinkedList orderOfLists;

    ListManager(ListCC l) {
        // int xOffset = xOffsetIn;
        // int yOffset = yOffsetIn;
        xMax = 0;
        yMax = 0;
        tList = l;
        current = tList;
        cu = tList;
        listMatrix = ListMatrix.emptyMatrix(MAX,MAX);
        orderOfLists = new LinkedList();
    }

    public ListCC getListCC() {
        return tList;
    }

    public void setOffsets(int a, int b) {
        xOffset = a;
        yOffset = b;
    }

    public void manage() { // Creates the initial matrix and manages the collisions
        until there is no more.

        createInitMatrix(current,0,0,null);
        // TextIO.putln(orderOfLists.toString());
        // listMatrix.printMatrix();
        while (!(detectCollision() == null)) {
            shiftListCC(detectCollision());
            //TextIO.putln(orderOfLists.toString());
            //listMatrix.printMatrix();
        }
        setManagedTrue(cu);
    }
}

```

```

public void setManagedTrue(SExpression cu) {
    if (cu != null) {
        cu.managed = true;
        if (cu.getClass().getName().equals("ListCC")) {
            ((ListCC) cu).name_obj.managed = true;
            if (!(((ListCC) cu).isEmpty()))
                setManagedTrue(((ListCC) cu).getFirst());
            // setManagedTrue(((ListCC)cu).car());
            // setManagedTrue(((ListCC)cu).cdr());
        }
        if (cu.getClass().getName().equals("Cons-cell")) {
            setManagedTrue(((ConsCell)cu).getCar());
            setManagedTrue(((ConsCell)cu).getCdr());
        }
    }
}

public ListMatrix getListMatrix() {
    return listMatrix;
}

public boolean samePos(SExpression a, SExpression b) { // return true if two
Cons-cells and/or SEInts have the same position
    return ((a.getX() == b.getX()) && (a.getY() == b.getY()));
}

public void shiftListCC(SExpression l) { // Shifts (in the matrix and the
canvas) a whole list "l" down one spot.
// that is, it moves all the elements of "l" one spot down.
if (l != null) {
    // if (l.getClass().getName().equals ("SEInt"))
    if (l.isAtom()) {
        shiftDownSingleExp(l); // shifts the SEInt
    } else if (l.getClass().getName().equals("ListCC")) {
        shiftListCCinMatrix(((ListCC) l));
        shiftDownSingleExp(l);
        shiftListCC(((ListCC) l).first());
    } else if (l.getClass().getName().equals("Cons-cell")) {
        shiftDownSingleExp(l);
        shiftListCC(((ConsCell) l).getCar());
        shiftListCC(((ConsCell) l).getCdr());
    }
}
}

public void shiftListCCinMatrix(ListCC l) {
    // Moves all the elements "list l" in position aij to the position a(i+1)j.
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            if (listMatrix.get(MAX-i-1,MAX-j-1).size() > 0) {

```

```

        if (listMatrix.get(MAX-i-1,MAX-j-1).contains(l)) // have to do
it from bottom up
        {
            listMatrix.get(MAX-i-1,MAX-j-1).remove(l);
            if (!(listMatrix.get(MAX-i-1+1,MAX-j-1).contains(l)))
                listMatrix.get(MAX-i-1+1,MAX-j-1).addFirst(l); // or
addFirst()?? or add()
        }
    }
}
}
}
}
}
}
}
}

```

```

public void shiftDownSingleExp(SEExpression s) {
    // Shifts an object (cons-cell or SEInt) down by xStep units down in the
canvas.
    s.setY(s.getY()+xStep);
    if (s.getY() > yMax)
        yMax = s.getY(); // update the yMax;
}

```

```

public ListCC detectCollision() {
    // First detects a collision in the matrix (that is where there is more
than one element,
    // in other words where more than one pointer to a list is found). Then
returns the list to be shifted down.
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            if (listMatrix.get(i,j).size() > 1)
                return (ListCC) giveListCC(listMatrix.get(i,j),i,j);
        }
    }
    return null;
}
}

```

```

public ListCC giveListCC(LinkedList l,int x, int y) {
    // Returns the list that needs to be shifted down.
    if ((l.size() > 2) || (detectType(l,x,y) == 0)) {
        for(int i = 0; i < orderOfLists.size(); i++) {
            for (int j = 0; j < l.size(); j++) {
                if (((ListCC)orderOfLists.get(i)) == ((ListCC)l.get(j)))
                    return (ListCC) l.get(j);
            }
        }
    }
    } else// collision between SEInt and cons-cell
    {
        if (detectType(l,x,y) == 1) {
            return (ListCC) l.get(1);
        }
        if (detectType(l,x,y) == 2) {
            return (ListCC) l.get(0);
        }
    }
}

```

```

    }
}

PVTSFrame.put(" this should never happen!");
return null;

}

public int detectType(LinkedList l, int i, int j) { // detects the type of
collision found, that is collisions between cons-cells and/or SEInts.
ListCC temp0 = (ListCC) l.get(0);
ListCC temp1 = (ListCC) l.get(1);
if (thereIs(temp0,i-1,j)) // then temp0 is an SEInt
{
    if (thereIs(temp1,i-1,j)) // temp1 is SEInt
    {
        return 0; // both are SEInts
    } else
        return 1; // temp0 is SEInt and temp1 is cons-cell
} else // temp0 is a cons-cell
{
    if (thereIs(temp1,i-1,j)) // temp1 is SEInt
    {
        return 2; // temp0 is a cons-cell and temp1 is an SEInt
    } else
        return 0; // both are cons-cells.
}
}

public boolean thereIs(ListCC list, int i, int j) { // returns true if "list" is
in the linkedlist of lists at position ij in the matrix.
ListIterator lI = listMatrix.get(i,j).listIterator();
while (lI.hasNext()) {
    ListCC l = (ListCC) lI.next();
    if (l == list)
        return true;
}
// TextIO.putln("could not find the list in "+i+", "+j+. ");
return false;
}

public int giveMaxY() {
    return yMax;
}

public void createInitMatrix(SExpression c, int x, int y, ListCC currentList)
{ // Creates the initial matrix and sets the appropriate coordinates to the lists
  (and its elements)
    // according to the initial matrix. This results in the visualization of a
  list that isn't
    // concerned about collisions.
    if (((ListCC) c).isEmpty()) {
        c.name_obj.setXY(yStep*y+y0Offset,xStep*x+x0Offset); // set the pos of

```

```

the name of the list
    c.setXY(yStep*y+yOffset,xStep*x+xOffset); // set the pos of the list.
    if (c.getY() > yMax)
        yMax = c.getY();
} else {
    while (c != null) {
        if (c.getClass().getName().equals("ListCC")) {
            currentList = (ListCC) c;
            // if (!((ListCC)c).isEmpty())
            // {
            if (((Conscell) ((ListCC)c).first).managed == true) {
                c.setXY(((Conscell) ((ListCC)c).first).getX(),((Conscell)
((ListCC)c).first).getY());
                c.name_obj.setXY(yStep*y+yOffset,xStep*x+xOffset); // set
the pos of the name of the list
            } else {
                c.name_obj.setXY(yStep*y+yOffset,xStep*x+xOffset); // set
the pos of the name of the list
                c.setXY(yStep*y+yOffset,xStep*x+xOffset); // set the pos of
the list.
            }
            if (c.getY() > yMax)
                yMax = c.getY();
            // TextIO.putln("the object: " + c + "has position: " +
c.getX() + ", " + c.getY());
            c = ((ListCC)c).first;
        }
        // }
        listMatrix.get(x,y).add(currentList); // add the current list in
the correct spot in the matrix
        if (!(orderOfLists.contains(currentList)))
            orderOfLists.add(currentList);
        c.setXY(yStep*y+yOffset,xStep*x+xOffset); // have to invert and y
because in matrix and graph coordinates its inverted
        if (c.getY() > yMax)
            yMax = c.getY();
        // TextIO.putln("the object: " + c + "has position: " + c.getX() +
", " + c.getY());
        // if (isAtom(((Conscell) c).getCar()))
        // if (((Conscell) c).getCar().getClass().getName().equals
("SEInt") || ((Conscell) c).getCar().getClass().getName().equals ("SEBool"))
        if (((Conscell) c).getCar().isAtom()) {
            listMatrix.get(x+1,y).add(currentList);
            if (!(orderOfLists.contains(currentList)))
                orderOfLists.add(currentList);

((Conscell)c).getCar().setXY(yStep*(y)+yOffset,xStep*(x+1)+xOffset);// have to
invert and y because in matrix and graph coordinates its inverted
            // TextIO.putln("the object: " + ((Conscell)c).getCar() +
"has position: " + ((Conscell)c).getCar().getX() + ", " +
((Conscell)c).getCar().getY());
        } else {
            if (((Conscell)

```

```
c).getCar().getClass().getName().equals("ListCC"))
    {
        if (((ListCC)((Conscell) c).getCar()).isEmpty()) {
            listMatrix.get(x+1,y).add(currentList);
            if (!(orderOfLists.contains(currentList)))
                orderOfLists.add(currentList);

            ((Conscell)c).getCar().setXY(yStep*(y)+yOffset,xStep*(x+1)+xOffset);
        } else
            createInitMatrix(((Conscell)
c).getCar(),x+1,y,currentList);
        } else
            createInitMatrix(((Conscell)
c).getCar(),x+1,y,currentList);
        // TextIO.putln("the object: " + ((Conscell)c).getCar() +
"has position: " + ((Conscell)c).getCar().getX() + ", " +
((Conscell)c).getCar().getY());
    }
    c = ((Conscell) c).getCdr();
    y++;
}
}
}
```

Function-calls trees display algorithm

The following java class (FunctionManager.java) contains the algorithm used to display the function-calls trees.

```

/*
 * FunctionManager.java
 *
 * Created on August 30, 2006, 7:36 PM
 *
 * author: David A. Pilo Mansion
 *
 Pilo's Visualization Tools for Scheme (PVTS). A Basic Visual Scheme Interpreter.
 Copyright (C) 2007 David A. Pilo Mansion

 This program is free software; you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation; either version 2 of the License, or
 (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License along
 with this program; if not, write to the Free Software Foundation, Inc.,
 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

 Contact information:
 David Pilo
 davidpilo@gmail.com
 */
import java.util.*;
import java.awt.*;
import javax.swing.*;
/**
 *
 * @author David Pilo
 */
public class FunctionManager {

    //public static int shiftConstant = 100;
    /** Creates a new instance of FunctionManager */
    public FunctionManager() {
        theTree = new ArrayList();
        for (int i = 0; i <50 ; i++) {
            this.theTree.add(i,new LinkedList());
        }
    }
}

```

```

    public void add(ProcedureAndArgs p, int i, boolean isdummy) {
        if (!isdummy)
            this.theTree.get(i).add(p);
        else
            this.theTree.get(i).add("dummy");
    }

    public ProcedureAndArgs getParent(int i, int j) {
        int numOfdummies = 0;
        for (int k = 0; k < j; k++) {
            if
(!this.theTree.get(i).get(k).getClass().getName().equalsIgnoreCase("ProcedureAndArg
s"))
                numOfdummies ++;
        }
        int temp = 0;
        int dum = numOfdummies;
        for (int l = 0; l <= dum; l++) {
            if (!this.theTree.get(i-
1).get(l).getClass().getName().equalsIgnoreCase("ProcedureAndArgs")) {
                temp ++;
                dum++;
            }
        }
        // PVTSTFrame.put("\nparent of: (" + i + ", " + j + ") is ");
        // PVTSTFrame.put("(" + (i-1) + ", " + numOfdummies + " + " + temp + ")");
        return ((ProcedureAndArgs)this.theTree.get(i-1).get(numOfdummies+temp));
    }

    public static void drawArrow(Graphics2D g2d, int xCenter, int yCenter, int x,
int y, float stroke) {
        double aDir=Math.atan2(xCenter-x,yCenter-y);
        g2d.drawLine(x,y,xCenter,yCenter);
        g2d.setStroke(new BasicStroke(1f));
        Polygon tmpPoly=new Polygon();
        int i1=12+(int)(stroke*2);
        int i2=6+(int)stroke;
        tmpPoly.addPoint(x,y);
        tmpPoly.addPoint(x+xCor(i1,aDir+.5),y+yCor(i1,aDir+.5));
        tmpPoly.addPoint(x+xCor(i2,aDir),y+yCor(i2,aDir));
        tmpPoly.addPoint(x+xCor(i1,aDir-.5),y+yCor(i1,aDir-.5));
        tmpPoly.addPoint(x,y);
        g2d.fillPolygon(tmpPoly);
        // g2d.setColor(Color.black);
        // g2d.drawPolygon(tmpPoly);
    }
    private static int yCor(int len, double dir) {return (int)(len *

```

```

Math.cos(dir));}
    private static int xCor(int len, double dir) {return (int)(len *
Math.sin(dir));}

    public void drawBranch (Graphics g,ProcedureAndArgs child, ProcedureAndArgs
parent) {

g.drawLine(child.xCoor+(child.width)/2,child.yCoor,parent.xCoor+(parent.width)/2,pa
rent.yCoor+parent.height);
    int xend = child.xCoor+(child.width)/2;
    int yend = child.yCoor;
    int xCenter = parent.xCoor+(parent.width)/2;
    int yCenter = parent.yCoor+parent.height;
    double aDir=Math.atan2(xCenter-xend,yCenter-yend);
g.drawLine(xend,yend,xCenter,yCenter);

    Polygon tmpPoly=new Polygon();
    int i1=12;//+(int)(stroke*2);
    int i2=10;//+(int)stroke;
    tmpPoly.addPoint(xend,yend);
    tmpPoly.addPoint(xend+xCor(i1,aDir+.5),yend+yCor(i1,aDir+.5));
    tmpPoly.addPoint(xend+xCor(i2,aDir),yend+yCor(i2,aDir));
    tmpPoly.addPoint(xend+xCor(i1,aDir-.5),yend+yCor(i1,aDir-.5));
    tmpPoly.addPoint(xend,yend);
    // g.drawPolygon(tmpPoly);
    g.fillPolygon(tmpPoly);
}

public void draw (Graphics g) {
    ProcedureAndArgs current;
    int temp_y = current_y;
    x = -1;
    y = -1;
    int max_x_panel = PVTSTree.ftrees.getSize().width - ((int)(100*zoom));
    //PVTSTree.put(max_x_panel);
    int num_of_calls;
    int coeff;
    for(int i = 0; i < this.theTree.size(); i++) {
        if (this.theTree.get(i).size() != 0) {
            y++;
            x = -1;
            for (int j = 0; j <this.theTree.get(i).size(); j++) {
                x++;
                if
(this.theTree.get(i).get(j).getClass().getName().equalsIgnoreCase("ProcedureAndArgs
")) {

                    current = ((ProcedureAndArgs)this.theTree.get(i).get(j));
                    if (viewMode ==3)
                        current.xCoor = (int) (80*x*zoom);
                    if (viewMode == 2)
                        current.xCoor = (int) (80*j*zoom);
                    if (viewMode == 0) {

```

```

        num_of_calls = this.theTree.get(i).size();
        coeff = max_x_panel / (num_of_calls);
        if (j == 0 && i == 0)
            current.xCoor = max_x_panel/2;
        else
            current.xCoor = coeff * (j+1);
    }
    if (viewMode == 1) {

        num_of_calls = this.theTree.get(i).size();
        coeff = max_x_panel / (num_of_calls);
        if (j == 0 && i == 0)
            current.xCoor = max_x_panel/2;
        else
            current.xCoor = coeff * (x+1);
    }
    current.yCoor = (int) ((90*y)*zoom)+ temp_y;
    current.width = (int) (70*zoom);
    current.height = (int) (25*zoom);
    if (current.yCoor > current_y)
        current_y = current.yCoor + 50;
    if (current.xCoor > max_x)
        max_x = current.xCoor + 50;
    g.setColor(Color.gray);

    g.fillRoundRect(current.xCoor+5,current.yCoor+4,current.width,current.height,20,20)
    ;

    g.setColor(DrawingTreesPanel.color);
    if (i == 0)
        g.setColor(DrawingTreesPanel.color.darker());

    g.fillRoundRect(current.xCoor,current.yCoor,current.width,current.height,20,20);
    //g.fillOval(current.xCoor,current.yCoor,current.width,current.height);
    g.setColor(Color.black);

    //g.drawOval(current.xCoor,current.yCoor,current.width,current.height);

    g.drawRoundRect(current.xCoor,current.yCoor,current.width,current.height,20,20);
    String s;
    if (funcNameDisplay)
        s = current.p.substring(0,1) +
current.sexprs.toString();
    else
        s = current.p + current.sexprs.toString();
    int l = s.length();
    if (l*7 > (current.width))
        s = s.substring(0,l-(l*7-current.width)/7);
    if ((current.result.isListCC() || current.result.isAtom())
&& result_bubble) {
        g.drawString(current.result.toString(), current.xCoor +

```

```

current.width + ((int)(30*zoom)) , current.yCoor+((current.height)/2)+6);
        Color temp = g.getColor();
        g.setColor(Color.red);
        Polygon p = new Polygon();

p.addPoint(current.xCoor+current.width+((int)(8*zoom)),current.yCoor+((int)current.
height/3));

p.addPoint(current.xCoor+current.width+((int)(14*zoom)),current.yCoor+((int)current
.height/3));

p.addPoint(current.xCoor+current.width+((int)(14*zoom)),current.yCoor+((int)current
.height/5));

p.addPoint(current.xCoor+current.width+((int)(25*zoom)),current.yCoor+((int)current
.height/2));

p.addPoint(current.xCoor+current.width+((int)(14*zoom)),current.yCoor+((int)current
.height*4/5));

p.addPoint(current.xCoor+current.width+((int)(14*zoom)),current.yCoor+((int)current
.height/3*2));

p.addPoint(current.xCoor+current.width+((int)(8*zoom)),current.yCoor+((int)current.
height/3*2));

        g.fillPolygon(p);
        g.setColor(Color.black);
        g.drawPolygon(p);
        g.setColor(temp);
        //g.drawLine(current.xCoor + current.width + 3,
current)
    }

g.drawString(s,current.xCoor+5,current.yCoor+((current.height)/2)+6);
    if (i != 0) { // draw child to parent branch if not root
        ProcedureAndArgs parent = getParent(i,j);
        drawBranch(g,current,parent);
    }
}
else {
    x--;
}
}
}
}
}

public ArrayList<LinkedList> theTree;
// public ArrayL
public int x;
public int y;
public static int max_x;

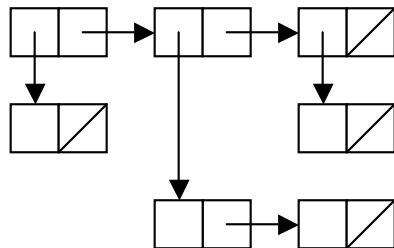
```

```
public static int current_y;  
public static double zoom = 1.0;  
public static int viewMode = 0;  
public static boolean funcNameDisplay = true;  
public static boolean result_bubble = true;  
  
}
```

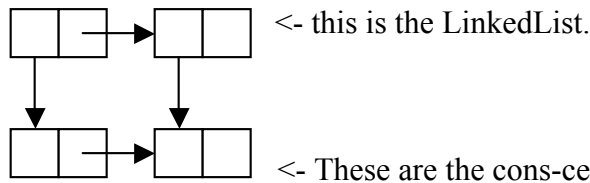

- Today I will try to design an algorithm that will keep track of the drawing depth as it draws. First will go through the whole list and then draw it.
- Scheme interpreter 1 – Problems:
 - o We can redefine a var, i.e: (define a 12) (define a 1) *this wasn't a problem actually scheme allows the user to do that.*
 - o We can do: (define a 12 1) when it should be an error.
 - o Big problem when entering: (list 2. Runs out of memory I have to test for bad inputs.
- Tomorrow Dr. Child and I will make a strategy to draw the cons-cells. My initial idea of keeping width and height of the cons-cells is good and will be used in that strategy.
- So far my interpreter accepts the following commands: list, define, car and +.
- I just run into an **IMPORTANT PROBLEM**, when trying to design the do_cdr method in the **interpreter** I realized that our data structure might not be the ideal one. I have to rethink how to store the lists (cons-cells). This also will impact the drawing **cons-cells program**.

5/23/06:

- Today I going to work with a new implementation of the cons-cells data structure. Working with lists (linked list) was not ideal.
- Working on Version 8 Cons-cells program.
- The new idea of representing cells is by storing the max depth and width of the cons-cells, i.e.: ((a) (b c) (d))



- A linked list simulation with cons-cells has been “implemented” without success really (this implementation can be found under **version 8**).
- The next step is to create a linked list that will contain as data the cons-cell it self (using the cons-cell class created in **version 8**). The new data structure may look like this:



- I am having LOTS of troubles with combining the new data structure for cons-cells with the interpreter (**version 3**).
- Designed a **COMPLETELY NEW** data structure for s-expressions, it can be found on the cons-cells program (**version 10**).

5/24/06:

- Today I am presenting to Dr. Child the new data structure designed yesterday.
- **Version 11** in the cons-cells program: the new (and definitive?) data structure (where cons-cells extend also Sexpressions) <- *this is an important version*.
- **Just got simpleparser3 to work!!!!** (just parsing not the drawing)
- **The drawing now works!!!** (although it draws list that are hardcoded)
- I need a **drawing manager** in order to tell the cons-cells lists and atoms where to in the canvas. **Version 12** also has the class separated into classes.
- Just started a **version 13** that will include the **manager**.

5/25/06:

- Important Note: in **version 13** we go back to make the cdr of type cons-cell.
- **Scheme interpreter** up and running on **version 4**.
- LEARNING EXPERIENCE: In scheme (list 2 3 4) will create a “temporarily-accessible” by the user reference (call it ref1) to a list containing the atoms 2, 3 and 4. In scheme (define list1 (list 2 3 4)) will bind list1 to ref1 by adding it into the symbol table (that is list1 will have the value ref1). In scheme (cdr list1) will create a “temporarily-accessible” by the user reference (call it ref2) to a list containing the atoms 3 and 4. In scheme (define list2(cdr list1)) will bind list2 to ref2 by adding it into the symbol table (that is list2 will have the value ref2).

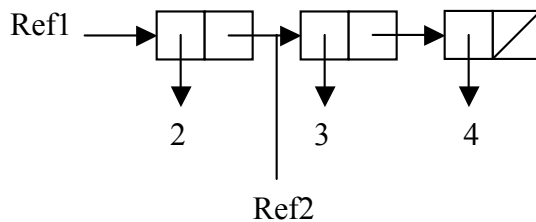


TABLE:
list1 = Ref1
list2 = Ref2

P.S: this whole thing came out after the examination of the function cdr() in ListCC.

- createInitMatrix now works! EXPLAIN HOW IT WORKS (MY OWN ALGORITHM)
- reArrangeNodes works! I can make the matrix that determines the position (**version 14 cons-cells program**).
- Need to find a way that during the matrix manipulations I change the position of the cons-cells.
- Got somethings drawing but not everythings works properly yet (**version 15**)
- It looks like the problem might be in the draw functions for some reason it doesn't draw a complete list.

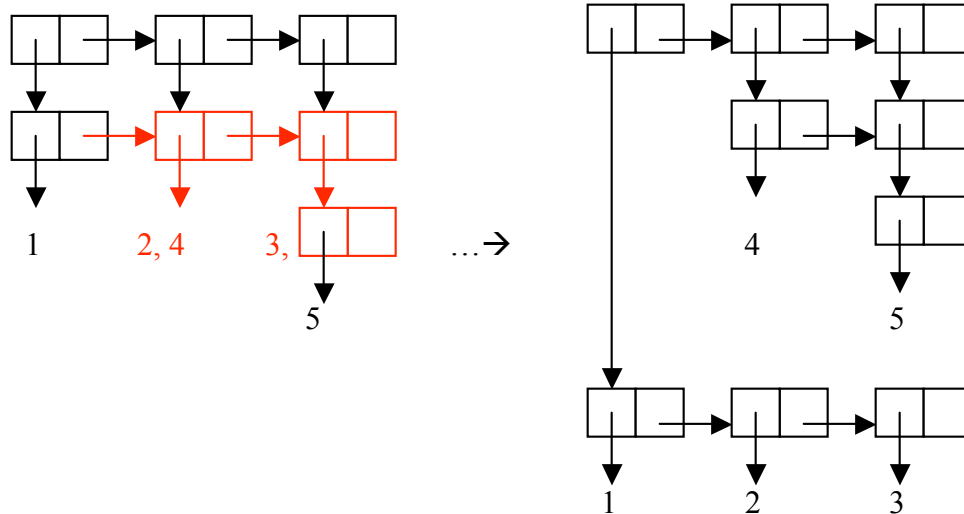
5/26/06:

- Idea to avoid collisions when drawing ListCC's: We will be using a matrix that will be used as a grid to place the cons-cells and atoms. An algorithm on the matrix will be performed so the final matrix contains no collisions.

- Algorithm: if there is a collision (that is there is an element greater than 1 in the matrix), subtract 1 to all the elements in the row where the collision happens. Add 1 to the elements in the next row. Repeat process until there are no collisions. The final matrix represents the new disposition of the cons-cells. Example:

((1 2 3) (4) ((5)))

1 1 1	1 1 1	1 1 1	1 1 1	...	1 1 1
1 2 2	0 1 1	0 1 1	0 1 1		0 1 1
1 2 2	2 3 3	1 2 2	0 1 1	...→	0 1 1
0 0 1	0 0 1	1 1 2	2 2 3		0 0 1
0 0 0	0 0 0	0 0 0	0 0 0		1 1 1
0 0 0	0 0 0	0 0 0	0 0 0		1 1 1



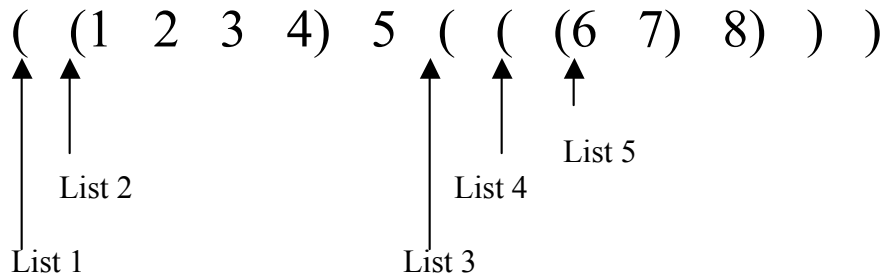
Note the collisions are in red.

- Problem with my algorithm is that the matrix doesn't tell the connections between nodes. Can I manage to make this algorithm work without using matrices, that is applying it directly to the cons-cells?
- FIXED draw functions in **version 15** (cons-cells program) and added a shiftListCC function that can shift a list down one spot (**version 16**)
- Going to work on a **version 17** add a class ListMatrix that will operate in matrices that contain a LinkedList of ListCC's.
- **Version 17** is SEMI-WORKING. Only specific types of lists won't display correctly. The problem is that I need to be able to decide which list will be shifted down in terms of "age" (that is which list has been "created" first).

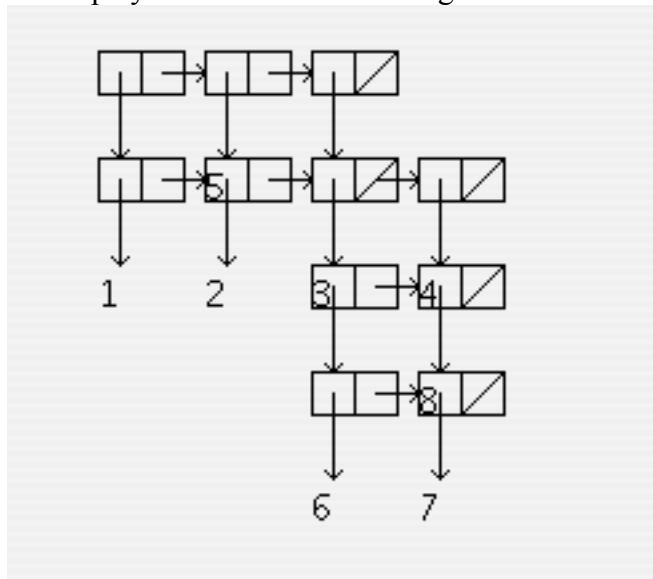
5/30/06:

- Trying different methods to get the correct shifting in order to have the correct display of the lists. Once I find the one that works I will explain the strategy used.

- **The most important day!!!! Version 18 works!** I can display ANY list! I tried this: ((1 1) (2 2) (3 3) (4)) (5 ((6 6) (7 7)) 8 ((9) 9)) 10)) and works fine!
- **Illustration of the algorithm:** Example list: ((1 2 3 4) 5 ((6 7) 8))



Given this list the algorithm will first create a matrix that will be used as a “grid” for the display of the list. The drawing with collisions will look like this:



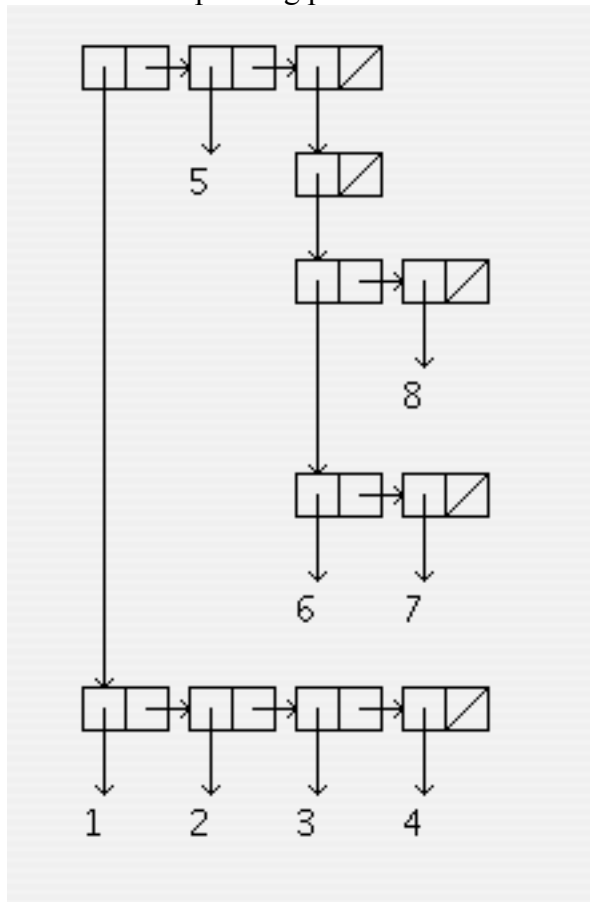
Hence the initial matrix will look like this:

1	1	1	
2	2,1	2,3	2
2	2	2,4	2,4
		5	5,4
		5	5

The algorithm will store in a matrix the lists. If an atom or a cons-cell belonging to list x is in the (i,j) spot then the reference to that list will be stored in the (i,j) element of the matrix. If there is a collision between an atom and a cons-cell, the list to which the cons-cell belongs to will be shifted down one spot, if the collision happens between two atoms or two cons-cells then the oldest list (the furthest left) will be shifted down. After running the algorithm (See appendix A to see the algorithm step by step) we obtain the final matrix:

1	1	1	
	1	3	
		4	4
			4
		5	5
		5	5
2	2	2	2
2	2	2	2

And the corresponding picture is:



- New goal: Merge the **interpreter** and the **cons-cell program**.
- Goal achieved it seems that **v1** works fine, the hybrid program seems to work fine.

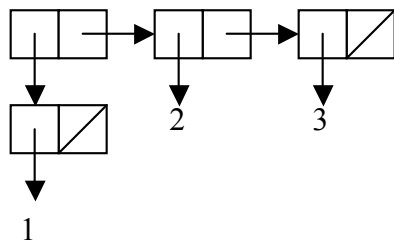
- Cleaned the code in **v1**. It would be nice to have the interpreter to give an option where the cons-cells can be displayed or not, so everything runs on a single executable.
- Reminder: finish the cons-cells program so I can output several lists on the screen.

5/31/06:

- New attempt: **v2.1** would consist of a single main program that would handle the interpreter and the drawing cons-cell. **V2.1** is somewhat functional.
- NOTE: I am not worrying too much about error detection. As long as the inputs are correct everything should work fine.
- Problem that I should solve some time: (list 1 2), doesn't work however (list 1 2) works (the problem is the blank space after the last argument. SOLVED (6/12/06)
- Right now I am working on error detection (**Interpreter**).
- I tried this example and worked beautifully:
(define yo (list (- 1) 2 3 (cdr (list 1 (car (list 1 2 (list 3))) (list (list 2) 3))))))
(vis yo)
- **V2.1** has a lot of error detection and has the following capabilities:
 - o car, cdr, list
 - o define,
 - o vis, vcar, vcdr,
 - o +, -
- Working on **v2.2** going to add new features.

6/1/06:

- Working on cons and vcons in **v2.2**.
- Cons works this way: (cons a l) will give a list whose first element is a then the rest of the elements are the elements in l, example : (cons (list 1) (list 2 3))

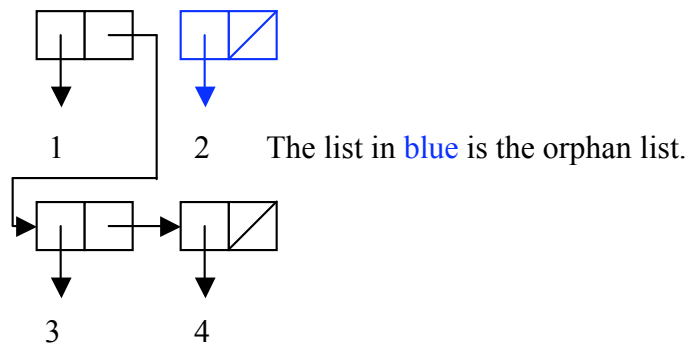


- Cons now works properly except that (cons 2 3) wont work because we haven't implemented the dotted pairs yet. **IMPLEMENTED**
- Problem: my switch in the interpreter doesn't catch the default statement, that is if I enter a word that is not a key word or a variable in the env it doesn't catch it, i.e. (yoyo 2) should say "function not found" and it doesn't. **SOLVED**
- Cons and vcons work, next step is set-car! And set-cdr! Functions.
- *Attention*: im going to use cons and *cheat* in order to implement set-cdr! And removeAllbutFrist() weird implementation.

- Set-car! And set-cdr! work properly except when trying to create dotted pairs.
IMPLEMENTED
- Working on a new version **v3** that will incorporate a way to read-eval-print multiple statements instead of just one statement. Hopefully in the future I will be able to read from a file (just an idea). Can't manage to do this as of today
- Going to add a * and / operations to the interpreter. **DONE** notice that the division will truncate my numbers so they are all integers (**v3**)
- Still trying to figure out how to be able to eval multiple expressions, will probably have to wait until dr. child comes back and talk to him about it.
- **TODO:** document the anti-collisions algorithm. **DONE.**
- **IMPORTANT:** I have to decide how am I going to handle the drawings when I execute commands such as set-cdr!.

6/5/06:

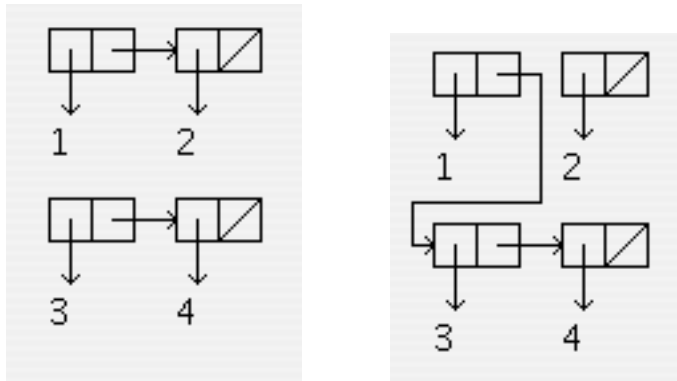
- To avoid terrible mistakes I am working on a new version (**v4.1**).
- Trying to implement the set-cdr! visualization.
- I am thinking about just having the vis command and get rid of the others.
- Going to attempt to fix the redraw in **v4.2**. **DONE**
- Fix the arrows when drawing the set-cdr! **DONE**
- New idea **orphan lists**. Example: (define a (list 1 2)) (define b (list 4 5)) (vis a) (vis b) (set-cdr! a b) should look like this:



- The set-cdr! now works with the orphan implementation. Example:

Before:

After:



- PROBLEM: I tried to color the orphan lists but it won't work for some strange reason, will have to talk about it with dr. Child. **SOLVED**
- Got **v4.1c** that accepts multiple expressions at a time.
- **V4.2** has the commands set-car! And set-cdr! working properly (including the visualization).
- **V4.3** combines v4.2 and v4.1c.

6/6/06:

- **V4.3** doesn't work, doing a new version **v5** in an attempt to merge multiple expressions at a time with the new features from v4.2.
- **V5.1** almost there. V5.2 closer.
- **V5.3** works! Finally! Most updated version as of 11:34 am. Now working only from this version.
- **V5.4** added the feature to draw empty lists.
- **V5.5** has labeled lists.

6/7/06:

- Problem when drawing many arrows there is some overlap. Trying to fix this with a random disposition of them. (v5.5).
- **V5.5** has the color problem fixed; the random generation for anti-collisions of arrows is still causing problems.
- Starting a new version **v6.1**
- **V6.2** if I add a textio output in the arrow class in the constructor, it messes up the outputs and get concurrent linked list modification exceptions.
- Still working on the set-car! Arrows drawing. DONE arrows are working now.
- V6.2 is a proof that there is two drawings!!! Check it! Have to talk about that with Dr. Child. That could be related to the problem I had with the concurrent exception.

6/8/06:

- I've discovered that my code has a problem: the lists are being drawn twice each time, the proof is that with the anti-collision of arrows we can see two arrows drawn! **SOLVED**
- For no apparent reason the arrows are not drawn twice?!? I have no clue why.
- **V6.3** has the cdr arrows fixed they are disposed randomly (like the car arrows)

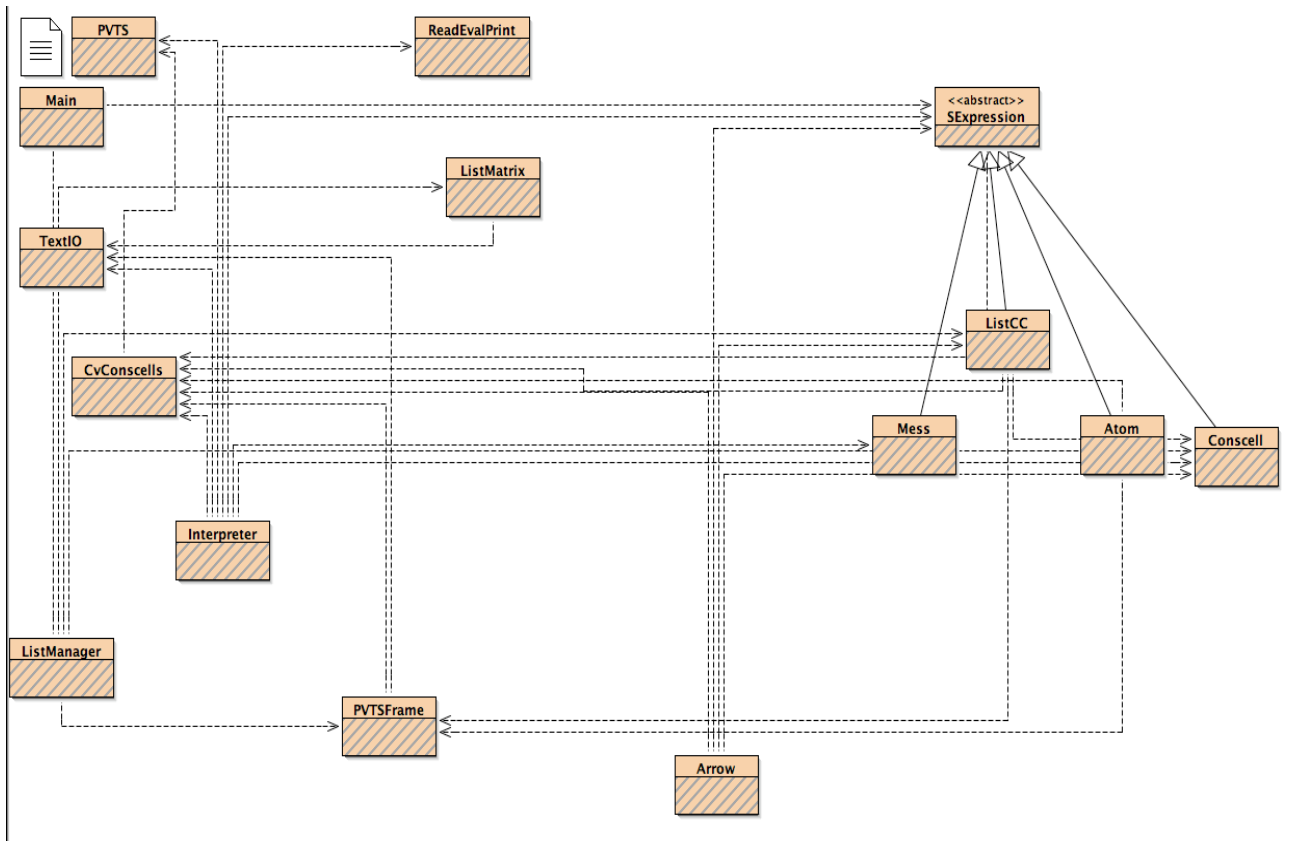
- Figure out how to send the text, instead of using the standard input. The buffer has to be replaced with the input obtained from the TextAreaInput.
- Working on a GUI. **V7.2** has a beta of this GUI.
- Now working a new version: **v7.3**.
- My GUI only works for one input, the next inputs won't work.

6/10/06:

- The GUI now accepts multiple inputs! The problem was that I wasn't resetting the position of the cursor when reading the buffer. (**v7.3**)
- Got rid of the command visu, and only have vis now. The command vis takes one or more lists and displays them.
- Have to start working on a PowerPoint presentation for Friday 16th june.
- Working on a new version (**v7.4**) trying to do the outputs in the GUI. DONE
- The program still has plenty of errors I will have to probably redesign the main structure of my program.

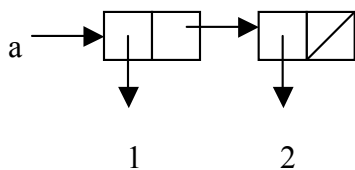
6/11/06:

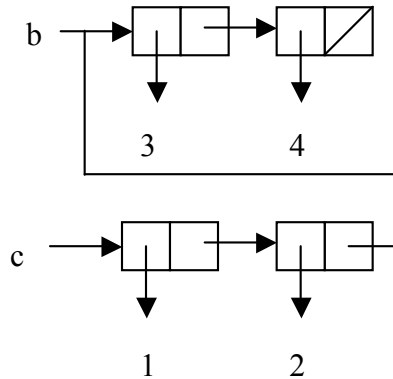
- "append" might still not work very well it produces errors, but maybe the errors come from somewhere else try this:
(define a (list 1 2 3))
(define b (list 4 5 6))
(define c (append a b))
(append)
(define d (append (list 1) (list 12 3) (list (list 1 2) 3)))
(vis a b c d)
- I have to think about the overall structure of my program and keep cleaning up.
- **RETHINK DESIGN!** I will use bluej in order to rethink my design.



6/12/06:

- New version **v8.1** with drastic design changes.
- I just solved the problem I had on the 5/31/06. Now things like (+ 3 4) will also work.
- When should the drawings be done? When doing a define? Should I keep the commands vis? Or get rid of it?
- When the user does a cdr should that be represented by highlighting the cdr of the list?
- PROBLEM: append works in a very special way, it copies all the lists except for the first one and then links them together. See appendix A for example/proof that illustrates this problem. **IMPLEMENTED/SOLVED**
- I need to implement my own needCopy since Java API does not provide and deep copy method only a shallow copy. I had no success in doing that so far... (v8.1)
- I just discovered how append really works: it will make a **deep copy** of all the lists received except for the last one, i.e. (define a (list 1 2)) (define b (list 3 4)) (define c (append a b)) will work as follows:





- PROBLEM: when redefining a var that is bound to a list to be another list, it will draw both of them! FIX THIS! **IMPLEMENTED/SOLVED**
- APPEND WORKS!!!! Except that I have to check with empty list and still the dotter pair problem. Also for some reason the PVTS>> output of the appended list doesn't print the last element of the list?? Weird!

6/13/06:

- Append now works and prints correctly (the problem was the size that wasn't updated correctly).
- PROBLEM: couldn't add to the canvas a scrollPane, online I have found that they suggest to not mix awt and swing, and use JPanel instead of canvas. I haven't figure out how I want to handle this situation yet. **IMPLEMENTED/SOLVED**
- Changed the appearance of my GUI **v8.1**. See appendix A to see what it looks like.
- Working on **v8.2** (made a slight change to the GUI).
- FIX: I can do (define 3 (list 1 2)) make sure the var always starts with a char not a number. **WRONG actually DrScheme lets me do that too!**
- Have to talk with dr Child why the list is not removed when redefined. What doesn't work is : env.get(id) ← FIXED! (I actually worked I had to remove from list first then from hashtable).
- Just added a reset button that basically erases all the data.
- Append had a problem with empty lists, now FIXED.
- BUG: try this: (define a (list 1 2 3)) (define b (list 3 4 5)) (set-car! a b) (define b (list 0 0 0)). There will be two lists b!!! instead of one. **SOLVED**

6/14/06:

- Just added the feature that if a list is redefined the previous one becomes orphan (**v8.2**)
- The list visualization (**v8.2**) is working pretty good, there is still many fixes to do but I will start working on the recursive calls program.
- IMPORTANT FIXES TO DO FOR THE LIST VISUALIZATION:
 - o Run button should reset too (like drscheme) (get rid of reset) DONE
 - o Maybe add a open file command? **IMPLEMENTED/SOLVED**
 - o The interpreter stops if there is a blank line between two statements so never gets to the second one. FIXED

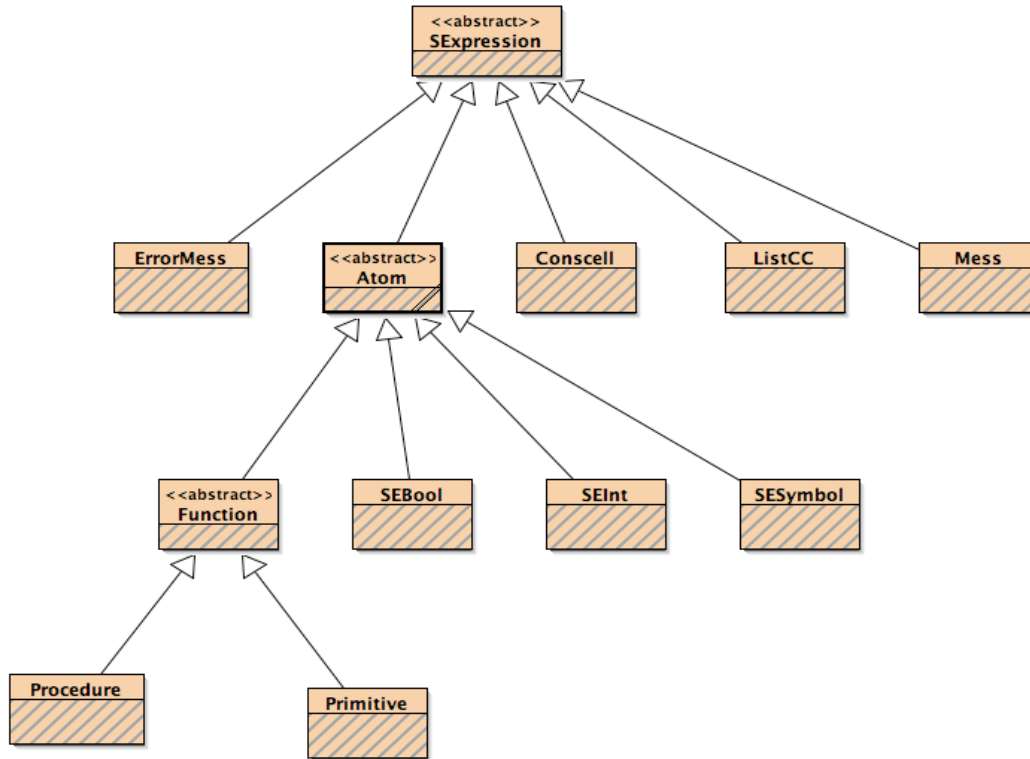
- Add command line functionality to the output dialog (like in drscheme).
 - Add functionality for dotted pairs **IMPLEMENTED/SOLVED**
 - Previous bugs, etc...
 - **Change the canvas for a Jpanel or something? Because canvas cannot have the scroll panel in it! Canvas is a heavyweight component (awt) and Jpanel is a lightweight component (swing). It would be better to not mix things. IMPLEMENTED/SOLVED**
- Recursive calls program starting at version: **v9.1:**
 - Run now resets before running.
 - Now my program visualizes EVERYTHING, got rid of the vis command.
 - NOTE: my program uses the character '~' to signal the end of the file. (**v9.2**)
 - New version **v9.3** I am going to add the define functions feature.

6/15/06:

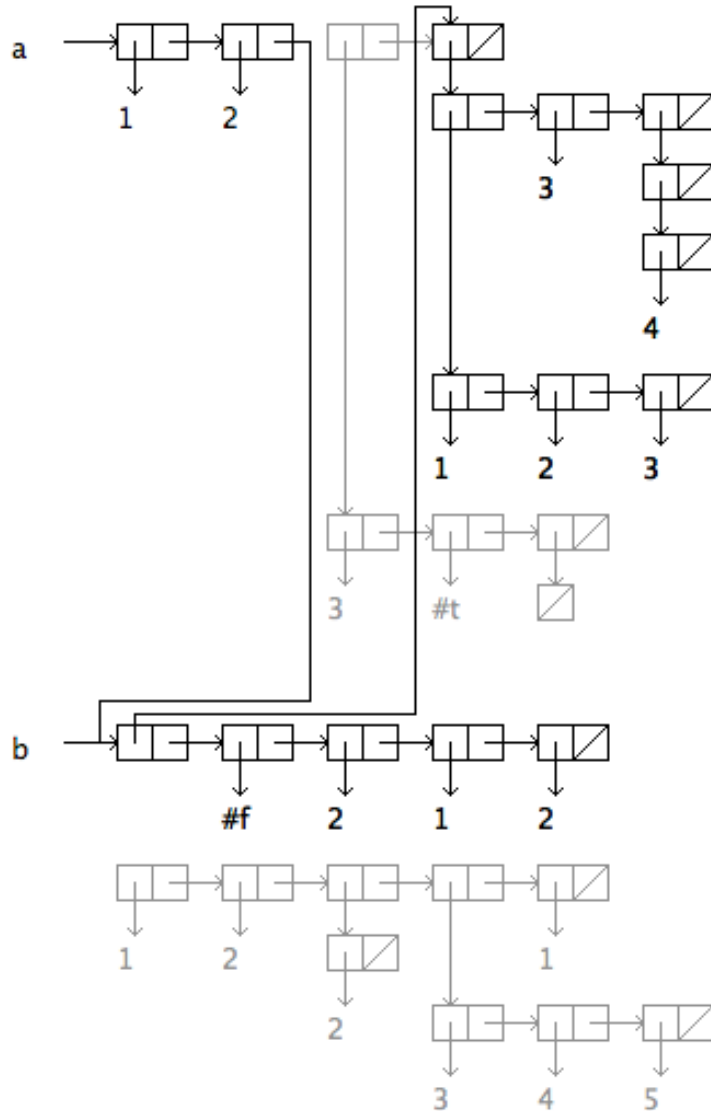
- **v9.4** has Booleans and operations on Booleans such as >, < and =. ☺
- **v9.5** I am going to make integers and bools be Atoms.
- PROBLEM: if doesn't work try this: (if (> 3 4) (list 1) (list 2)) shouldn't display both!!! If is a special form the parameters don't get evaluated until need.
IMPLEMENTED/SOLVED

6/16/06:

- new version **v9.7** I am going to go back to my way of visualizing things less memory-like and more user-friendly like. DONE
- new version **v9.8**. if works now properly.
- Check in the future for possible syntax errors (with "if" among other commands).
- Now we have new data structure (this was implemented in v9.4 I believe)



- “if” doesn’t work: try this: (if (= 3 4) (list 1 2) 1), SOLVED
- “if” should work now properly.
- New version **v9.9** I will introduce a arrow data base that will take care of the multiple arrows problem. This will make sure that if there is an arrow between two objects the next arrow will be drawn exactly at the same spot to avoid multiple arrows.
- See appendix A for the only work that has a similarity with mine (its in lisp and its functionality seems to be very different)
- **V9.9 HAS FIXED THE PROBLEM OF MULTIPLE ARROWS!!** (v6.3 has a picture of the problem) this is how it looks like now:



- To solve this problem I have created and Arrow DataBase that keeps track of all the arrows already drawn so next time there is a arrow that will be drawn it will check first if there was an arrow already and if it is the case then it will use the same antiColVar.
- PROBLEM: if I do: (define a (list 1 2 3)) (define b a) it doesn't draw to pointers to the same list only the pointer of b is drawn. I haven't found a solution to that problem yet. SOLVE IT! This is important. **IMPLEMENTED/SOLVED**
- New version **v10.1** I am going to try to solve the last problem.
- "if" had more problems (fixed) a now I have a new GUI. (the text output is underneath the input. Now the canvas has more space down below).

6/17/06:

- I had a GENIUS idea, I am going to add a stepper! ☺ **v10.1(bis stepper)**. DONE!!
- Now working on new version **v10.2**. try to solve the multiple pointers to a list.
- PROBLEM: Something is still wrong with the “if” and also the comments only work when in stepper mode I don’t know why. Fixed the comments in the run mode and the if has a problem, I don’t parse the last parens try this: (if (> 3 2) 2 1) check also other’s expressions parens. Drscheme gives: read: expected a ')' error. FIXED

6/18/06:

- **V10.2** is pretty solid. (comments on stepper and run work. “if” works too) (this version was sent to dr.child)
- Starting a new version: **v10.3**
- The problem is a list cannot a a single name. Because a and b can point to the same list. Discovered this in **v10.4 with SENAME**
- Starting a new version to solve the problem, from v10.3: **v10.5**

6/19/06:

- saved a backup **v11.1**.
- saved a backup **v11.2 I am done with cons-cells** for now!
- Working on the quote now.
- QUOTE almost works!!!! What doesn’t work yet is ‘ab should print ‘ab instead of just ab.
- Also work on quoting operations such as ‘(+ 2 3). That is next.
- Quote should work
- I have again the append problem I had before (6/12/06) I am going to solve it (the problem was the size wasn’t updated correctly. FIXED the size was incorrect in “copy” and “append”!
- IMPORTANT THOUGHT: I think I should insert primitive type when loading primitives instead of using SEInts. I could insert primitives and when the user defines it would be a function. I tried and failed!
- Done a few changes in the rep() func and catch the BUG that would result in an infinite loop ☺. Back up done at **v11.5**
- **I am able to define procedures!!!**

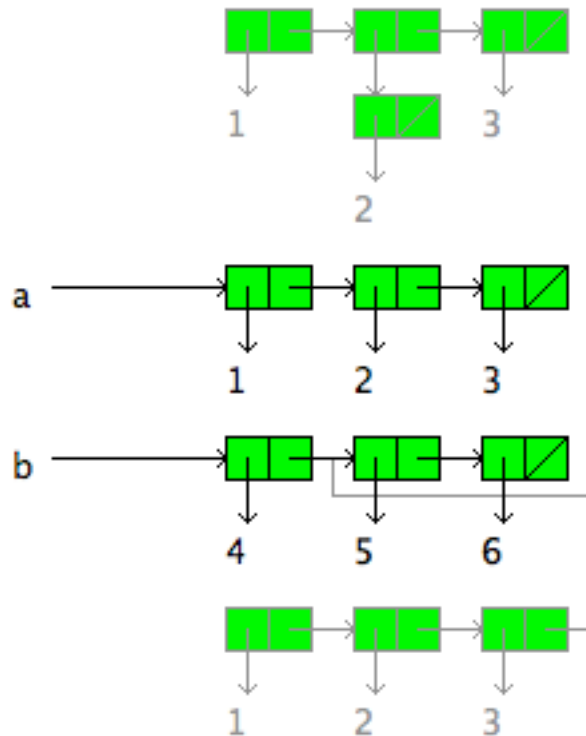
6/20/06:

- Talk to child about my current define_func, see what he thinks.
- **Backup v11.6**. I am able to define functions not run them though
- Goal to evaluate user-defined functions (also called procedures).
- I have just implemented the Environment. It is a stack of hashtables. Each hashtable is a “local environment” it stores the variables, procedures and primitives. **Backup v11.7**.
- This is how a functions get evaluated:
 - First push a new environment
 - Define in the enviroment the formal parameters to be the variables passed to the function
 - eval the body. (which is evaluating a list...)

- I have been able to catch the error when the number of parameters sent to the procedure is not the same number of formal parameters expected ☺
- Fix this: (list - 2 3) '(- 2 3) outputs should be different. **I decided to keep printing <primitive:-> which is more informative than just a minus sign.**
- Obviously at this moment I can't get the evaluation of a function to work yet, but I am progressing very fast.

6/21/06:

- I got simple functions to work. I.e.
;define functions tests:
(define (addone x) (+ x 1))
(addone 1)
(addone 4)
(define (mysquare y) (* y y))
(define (myadd x y) (+ x y))
(mysquare 7)
(myadd 3 4)
- That test works fine! ☺ (backup **v11.8**)
- In scheme a only the last Sexpression is printed in a function definition. My function evaluation doesn't do that yet. **FIX THIS EXAMPLE:**
- (define (def x) (define yo 50) (display "yo!") 2 (+ 50 yo))
- (def 1) should print yo!100. (so only the last exp is printed however everything is evaluated.) **FIXED**
- I have a new little problem when running a code and the pressing the stepper, the program gets kind of confused (only when evaluating procedures). Fixed on v12.1 ☺
- Another issue: I just discovered that the body of a function is a collection of Sexpressions not just one. I am going to fix this and the latter problems should be fixed with some modifications I just did. (backup **v11.9**)
- **VERY IMPORTANT STEP AND VERSION FUNCTIONS NOW WORK!!! ALL PREVIOUS PROBLEMS ON FUNCTIONS SOLVED.** (backup **v12.1**)
- My environment was looking for variables from the bottom up instead of up to bottom that caused my hours of debugging.
- Factorial TEST WORKS!!!! THIS IS HUGE
- FIBONACCI TEST WORKS!!! THIS IS UNBELIEABLE!! I AM A MAC GENIUS. (BACKUP **v12.2**).
- I should implement "cond". **DONE**
- Check the code commented in draw cons-cells because drawing green rectangles doesn't work!?! Its my fill rect call.
- **FIX THIS:** if the doc finishes with a comment the interpreter returns an error unexpected '~'. **FIXED**
- Just added colors to the cons-cells:



-
- Working on “cond” right now. Doing tests to figure out how it really works.


```
(define (foo x) (cond
  ((= x 1) (display "yo") 2 #f)
  ((= x 2) 3 4 5)
  (else (- 3 4))))
(foo 1)
(foo 2)
(foo 3)
```
- will return :


```
Welcome to DrScheme, version 301.14-svn12may2006.
Language: Standard (R5RS).
yo#f
5
-1
```
- So in a way it is very similar to what procedures do, only the last exp gets returned.
- FIX cond, it is still quite not working properly.

6/22/06:

- I am going to fix cond. DONE!
- I just added to getWord() to stop when encountering a double quote “ (also added to anotherexpr()). Because eventually I want my interpreter to accept Strings as atoms. DONE
- I would like to add also a display function. DONE

- In my cond I had a problem with the else because I can't do a peek word. I solved by creating my own peek word function and added to TextIO.
- I am adding "and". Or, not, display primitives.
- Things to be added or fixed
 - o Scheme is not case sensitive (try (And)) it should work but doesn't
 - o Add a matching parenthesis in my intextArea DONE
 - o Add a highlight feature in the inTextArea when the stepper is on to know where I am when stepping. DONE
 - o Use arbitrary precision integers. (big integers in java) NOT CONSIDERED
- Little problem (display "yo!") works fine however if I have (display (list "yo!")) it doesn't print like drscheme. (backup v12.4)
- Working on the "let" right now.
- CHECK if every time I push an env, I am popping it. CHECKED
- Let is done! (backup v12.5)
- Implementing "null?" done.
- Added in the name of the variables the environment to which they belong. (backup v12.6)
- I am going to implement assoc
- I have implemented "assoc" and "equal?" (backup v12.7).

6/23/06

- Caught the error that such that (let ((x 1)) (x));
 - FIX this: (list 1 2 or '(1 2 needs to throw error expected ')'. FIXED
 - Fix (let ((x '(1 2)) (y '(3 4))) (append x y)) doesn't display environment belonging
 - Fix infinite loop try this: (let ((x 1) (y 2)) x y) FIXED
 - Fixed an important error with environment get and contain (would only look at the first environment). FIXED!
 - FIX this (is not working)


```
(define (qsort2 lis order)
  (cond
    ((null? lis) '())
    (else '(1))))

(qsort2 '(2) 3)
(qsort2 '() 3)
```
 - does an infinite loop at while ln 93 in the interpreter. (bugs at the else)
 - He returns the last error in the interpreter
 - And also says undefined var else!
- HERE IS THE ERROR: (define (insertionSort lis) (cond ((null? lis) lis) (else (insertElmt (car lis) (insertionSort (cdr lis)))))) works: but this doesn't:
- ```
(define (insertionSort lis)
 (cond
```

```
((null? lis) '())
(else (insertElmt (car lis) (insertionSort (cdr lis))))))
```

- THE ERROR IS WHEN THE COND PART IS A LIST INSTEAD OF AN OPERATION. EX ((null? lis) '()) DOESN'T WORK HOWEVER ((null? lis) lis) works.
- If already in quote mode putting a '() will enter quote mode and exit before it should. SO inside a procedure I still can figure out how to use quotes... big problem, also check append sometime... (I modified do\_quote()) and added the "DANGEROUS CODE" (backup v12.8) **fixed. NOT SO DANGEROUS NOW ☺**

6/24/06

- I have to check my append do crazy tests with it. (append seem to work at least outside a procedure)
- I think I have nailed a problem: the problem is to use inside a procedure the "append" command with a quoted list.
- With david\_hw3 I get again the infinite loop while line 93 in the interpreter....
- This is what happens: if it was a quote I would be safe.

```
(define lis (list (3 4) 2 1 1 ((4) 4))) (cond ((null? lis) (list)) ((not (isIn (car lis) (cdr lis)))
(cons (car lis) (convertListToSet (cdr lis)))) (else (convertListToSet (cdr lis))))
```

- I had problems with case sensitive words, I made everything caseinsensitive like scheme does.
- This drove me nuts! My not function was not negating!! Hours of debugging just for that!
- I should implement eqv? And eq? sometime. Done!
- Qsort example works now!!!!!!!

FIX THIS: ((null? lis) (list)) ← this works but this doesnt → ((null? lis) '()) (inside a procedure definition. IMPORTANT BUG FIXED on the 7/24/06.

- BACKUP (probably the most important so far). **V12.9**

6/25/06:

- implement "list?". DONE
- fix this: (equal? 'a 'a) (prints "#t instead of #t). **done**
- in hw2chap15 for some reason the last teststruct doesn't work. (find out why) problem (backup v14.1). FIXED the problem was that the or was not working properly. (7/19/06)

6/26/06:

- I am modifying getWord in TextIO so it follows this def: NAME\_A name under a quote corresponds to a symbol. A name is a sequence keyboard characters, not including the following:

```
space " , ' ` () [] { } | ; #
```

- This doesn't work: `;(define (foo x) (if (= x 1) #t #f))`
- `;(foo 1)`
- `;(foo 2)`
- Very important: A parenthesized sequence of quoted elements abbreviates a list of elements. For example, `'(a b)` abbreviates `(list 'a 'b)`.
- 

6/28/06:

- Just implemented the methods: “odd?” and “even?”
- I am going to attempt adding SEFloat so I am going to make a class SENumber and SEInt and SEFloat will extend that class
- I have added floats however now I will have to change many of my functions so it can read both types. I think I should be smart about it and make general methods so I can be easily inserted in the already existing code.

7/3/06:

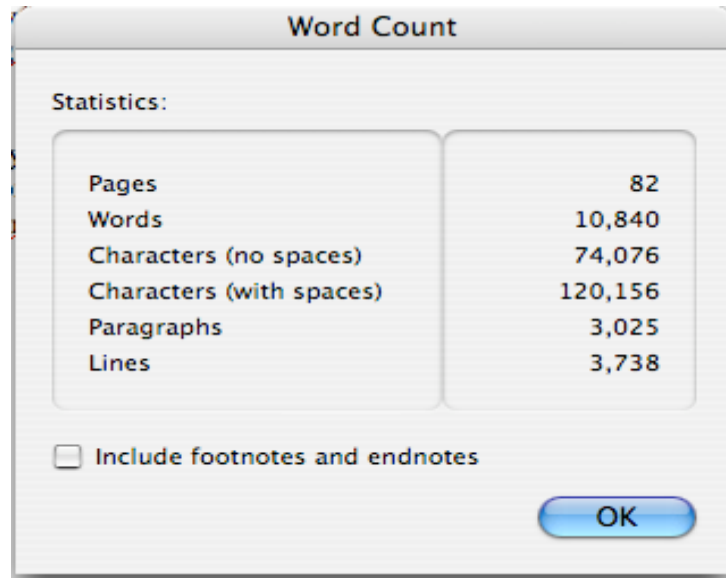
- Modified addition, multiplication and division in order to support both integers and reals.
- Note that my program if working with integers will not use fractions as opposed to drscheme. It will truncate, i.e, `(/ 3 2)` will return 1 instead of the fraction of 3 over 2.
- Working on making comparisons (`>` etc..) work for ints and reals.

7/5/06:

- new backup **v14.3**.
- I tried to implement the PVTS using Integer and Double as subclasses of Number without success I will save this version though under **v14.4 no success**.
- Came back to version v14.3

7/7/06:

- caught a new bug, I could not enter negative numbers . FIXED
- Adding abs, cos, sin, tan, acos, asin, atan, sqrt functions. DONE!
- Watch this:
- `(max 1 1 -2 0.34 45.2)`
- `(min 1 1 -2 0.34 45.2)` min returns `-2.0` (a float) when it was a integer!!! I don't do that though I return whatever type it was. Discuss with child.
- Added max and min functions.
- Quatity of work: as of today I am not counting TEXTIO.java since I haven't done much in that file, all the other files are from SCRATCH.



7/19/06:

- Fixed the teststruct problem (the or wasn't working as intended).

7/23/06:

- Fixed the use of quoting when applied to variables (symbols) for example 'a should return a instead of 'a.

7/24/06:

- Fixed the problem with having quoted lists inside the definition of a procedure.
- Added the commands eq? And eqv?. (Their implementation is so they call equal? In order to avoid complications.)
- Consider this problem: 23e in drscheme will say 23e is undefined, while my program prints 23 and then says e is undefined. FIX THIS? (Backup done at v14.7)

7/25/06:

- Currently implementing the set! Command. DONE
- If I want the example of the bank account to work I need to be able to do things like this: (define (s x) car)
- ((s 2) '(2))
- and right now I am not able. Backup at v14.8 in order to solve that.
- I got it to work now!!! Now I am capable of having functions that instead of returning data they return a function and then evaluates! ☺
- With this last version now I have created more problems than solved so I'm going to go back to v14.8.

7/26/06:

- Now this kind of things work: (I had to add the quote when passing a symbol as a parameter to a procedure i.e.:
- (define (yo x)  
     (if (equal? x 'hola) 1 0))

```
"xd"
(yo 'hola)
 (yo 'juas)
```

- Made small modifications to the code, to improve structure.
- Backup made **v15.1**.

7/27/06:

- added support for a procedure to be evaluated into another procedure. I previously had support for a procedure to evaluate into a primitive (as well as atoms and lists of course). Backup done at **v15.4**.

8/1/06:

- I have finally been able to remove the heavyweight component canvas and now I use a Jpanel instead of the canvas (so I use swing instead of awt). I also managed now to make it scrollable with a scroll pane ☺ (JScrollPane is only compatible with lightweight components –that is swing components). Backup at version **v15.7**.

8/3/06:

- Added tabs into my program! So I can draw the trees and the cons-cells in different tabbed Jpanels.
- FIX this important bug!: (define a (list 1 2 3))  
                                   (define b (list 2 3))  
                                   (define c b)  
                                   (define z (cdr c))  
                                   (define t (cdr c))  
                                   (define v z) FIXED!!!!!!!!!!!! on 9/6/06 fixed the arrow problem too.

- backup **v15.8**.

8/4/06:

- FIX this: power\_of\_recursion does not work if the name of the procedure appears lets say three times but the procedure is doubly recursive... when do I fix this? How?
- Latter problem fixed with completely new implementation of the solution to the problem.

8/7/06:

- first version of the recursive trees done. Backup version **v15.9**.

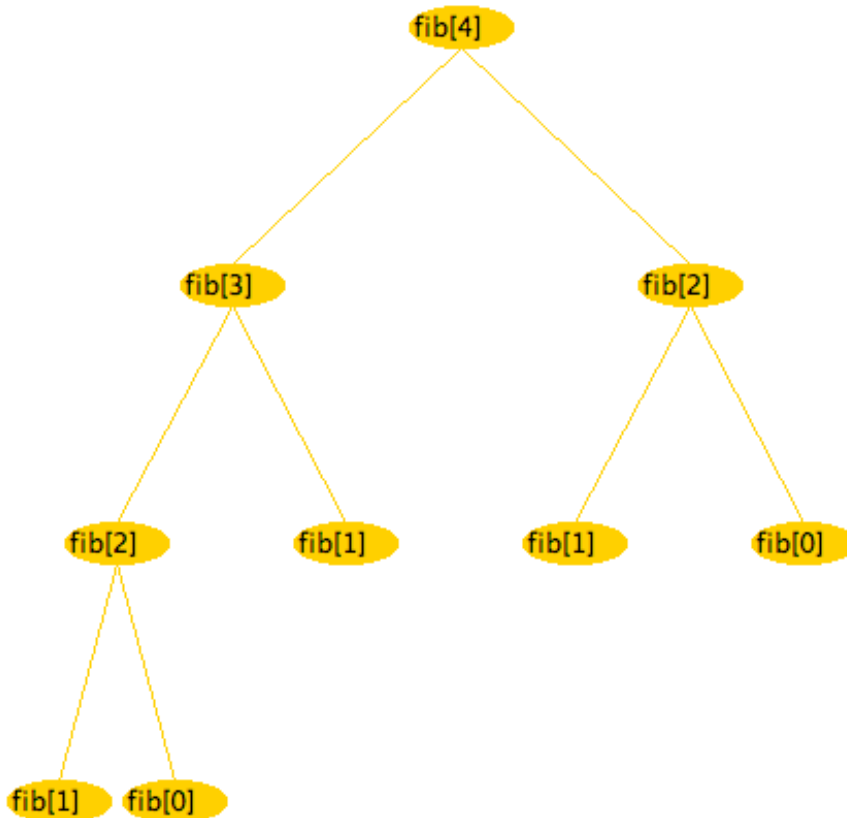
8/10/06:

- add this: (define (fib n)  
           (cond  
           ((= n 0) 0)  
           ((= n 1) 1)  
           (else (+ (fib (- n 1)) (fib (- n 2))))))

```
(define (g x) (fib x))
```

```
(g 5)
```

- the example above should include a circle having `g[5]` pointing down to the tree with `fib[5]`.
- Backup done at version **v16.1**. this is how a tree looks like as of version 16.1:



- now trying to rethink my problem: I want to actually display any procedure call (even primitives?) whether the procedure is recursive or not it should be displayed.

8/16/06:

- I have implemented the following prim-ops:
  - Exp
  - Expt
  - Positive?
  - Negative?
  - Ceiling
  - Floor
  - Integer?

- Real?
- Number?
- Modulo
- Procedure?
- log

8/25/06:

- new interface
- added options tab
- added a progressive zoom for functions display
- solved the windows-os problem when displaying the drawings.
- Backup version **v16.6**.

8/29/06:

- Given beta copy to Anderson **v16.8beta**.
- new GOALS:
  - change textArea to textformatted so I can use colors for errors and use highlights for matching parenthesis. DONE
  - Add open file, save file capabilities
  - Fix scolling
  - Options panel should represent current state and add a cancel button.
  - Represent any function calls

8/30/06:

- Added matching parenthesis capability.
- Improved matching parenthesis.
- Added a highlight when error is detected where the error is located.
- Started to rework the whole functions display from scratch. The previous idea had recursiveness as a property of each procedure, that was one of the main problems since now I want my program to display virtually any function call (recursive or not).

9/4/06:

- FIX this: (define a (cons 2 3))
  - a
  - (set-cdr! a '(4))
  - a
  - (define f (list 1 2 3))
  - f
  - (define yo (list 5))
  - (set-cdr! f yo)
- added support for DOTTED PAIRS.
- FIX this: '(3 4 . 5) should work. FIXED
- Backup **v18.5**.

- TOTALLY REVAMPED GUI see pic in appendix.

**9/5/06:**

- now dotted pairs fully supported!
- Added center alignment for function display.
- Backup version **v18.7**.

## 9/7/06

- Fixed bugs:
  - o Dotted pairs and appends
  - o Dotted pairs and set-cdr!'s
- Added the read command but still doesn't work because I don't know how to pause my program. Threading interrupt doesn't work.

## 9/12/06:

- Added support for Characters. Fixed bug with the if clause if else part was inexistent an error was thrown.
- Revamped the arrows, changed the look of the function tree calls.
- Added the result next to the oval of the function tree calls.

## 9/30/06:

- added support for lambda notation!!!!!! HUGEEEEEEEE STEP
- good error messages such as: (define yo (lambda x) x))
- (yo 6) will through three nice lines of errors.
- Added a popup window that displays information about the elements in the environment tables.
- Implemented the member prim-op
- Implemented list-ref prim-op

## 10/2/06:

- TO DO: implement begin. DONE
- Fixed major bug with the map functions (wasn't capturing procedures, only prim-ops where handled)
- Implemented GCD prim-op.

## 10/12/06:

- implemented the lamda functions with variable number of parameters feature for example:
  - (define yo (lambda x (cdr x)))
  - (yo 3)
  - (yo 4 5)

- returns () and (5)

10/24/06:

- working on the auto-indent (it almost works nicely!!!!) only need to make it work when I mouse click somewhere inside the code keep the auto indenting consistent.

10/26/06:

- auto-indent now is perfect. This time I use keyevent and only run the algorithm when the enter key is pressed. With this implementation I don't have the problems I had with the other version.
- I changed the implementation for division, now division ALWAYS outputs a float. (**v20.1**)

10/29/06:

- I have fixed the division prim-op. Before it would trim and return an integer if all numbers were integers. Now it will return real numbers, except if the result is supposed to be an integer (i.e. (/ 4 2) returns the integer 2. On the other hand, (/ 5 2) will return the float 2.5)

11/3/06:

- IDEA! WHEN CLICKING ON A TREE OR CONS-CELLS OR SEXPRESSION IN THE PROGRAM THEN HIGHLIGHT THE CORRESPONDING CODE, TREE AND LISTS! TO IMPLEMENT OVER THE BREAK. DECIDED NOT TO DO.

11/22/06:

- I have implemented the option to do NEW FILE, OPEN FILE, SAVE and SAVE AS... !!!! (**v20.3**)
- Do fractions? [http://mitpress.mit.edu/sicp/full-text/book/book-Z-H-14.html#%25\\_sec\\_2.1.1](http://mitpress.mit.edu/sicp/full-text/book/book-Z-H-14.html#%25_sec_2.1.1) DECIDED NOT TO DO,

12/11/06:

- Important change! Saved on version **v20.7** (**v20.6 is to revert, undo the change**) now Mess toString is implemented so it prints the body of the defined function, all other tests seem to work fine! No more return of "Sexpression!" so I have to see what could be the impact. So far it looks like it has no impact (all tests run fine).

12/17/06:

- **V21.1** Has many fixes, this version is HALF working with bank account. A lot of fixes on procedures that do not return Sexpressions are been made.

12/18/06

- Bank account problem almost done. Although I have a problem with the environments because the balance gets reset each time to the initial value.
- I have to REDO bank account problem, procedure should not redefine everything, a procedure should have a mini environment implemented inside and a body not just a body.

12/21/06:

- **V22.1 BANK ACCOUNT PROBLEM SOLVED!!!!**

12/23/06:

- Working on a new version in order to add a Recursive Functions Viewer that will display in text the functions. (v22.2).

**12/24/06:**

- Major version backup, with a new visualization module that shows a text version of the recursive calls. **V22.3**
- Implemented cut, copy, and paste actions for the input (and tabsize)

**12/26/06:**

- Implementing the following
  - o Char?
  - o String?
- On a previous version implemented highlight on current cons-cells when in stepper mode.

**1/17/07:**

- Added symbol? And for-each primary operators.
- Corrected many previous bugs that are mentioned in this log.
- Minor improvements in the GUI.

**1/18/07:**

- added indentation in the text version of the visualization of function calls.

**1/23/07:**

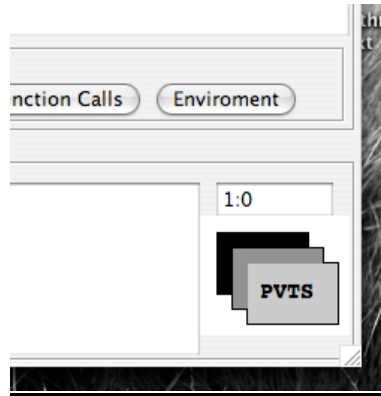
- Improved the highlights for the stepper mode, included close file menu
- Added a new menu, the examples menu that loads examples (list, append, bank account, Fibonacci and factorial examples) **v23.4**

**1/24/07:**

- added a logo on the main window the PVTS logo ☺ **v23.5**

**3/25/07:**

- Final version **v24.2**.



**1/30/07:**

- added GPL license final version v23.8

**APENDIX A of the WORK LOG:**

Step by step algorithm to get rid of the collisions:

|   |     |     |     |
|---|-----|-----|-----|
| 1 | 1   | 1   |     |
| 2 | 2,1 | 2,3 | 2   |
| 2 | 2   | 2,4 | 2,4 |
|   |     | 5   | 5,4 |
|   |     | 5   | 5   |

|   |   |     |       |
|---|---|-----|-------|
| 1 | 1 | 1   |       |
|   | 1 | 3   |       |
| 2 | 2 | 2,4 | 2,4   |
| 2 | 2 | 2,5 | 2,4,5 |
|   |   | 5   | 5     |

|   |   |     |       |
|---|---|-----|-------|
| 1 | 1 | 1   |       |
|   | 1 | 3   |       |
|   |   | 4   | 4     |
| 2 | 2 | 2,5 | 2,4,5 |
| 2 | 2 | 2,5 | 2,5   |
|   |   |     |       |

|   |   |     |     |
|---|---|-----|-----|
| 1 | 1 | 1   |     |
|   | 1 | 3   |     |
|   |   | 4   | 4   |
|   |   | 5   | 4,5 |
| 2 | 2 | 2,5 | 2,5 |
| 2 | 2 | 2   | 2   |

|   |   |     |     |
|---|---|-----|-----|
| 1 | 1 | 1   |     |
|   | 1 | 3   |     |
|   |   | 4   | 4   |
|   |   |     | 4   |
| 2 | 2 | 2,5 | 2,5 |

|   |   |   |   |
|---|---|---|---|
| 1 | 1 | 1 |   |
|   | 1 | 3 |   |
|   |   | 4 | 4 |
|   |   |   | 4 |
|   |   | 5 | 5 |

|   |   |     |     |
|---|---|-----|-----|
| 2 | 2 | 2,5 | 2,5 |
|   |   |     |     |

|   |   |     |     |
|---|---|-----|-----|
| 2 | 2 | 2,5 | 2,5 |
| 2 | 2 | 2   | 2   |

|   |   |   |   |
|---|---|---|---|
| 1 | 1 | 1 |   |
|   | 1 | 3 |   |
|   |   | 4 | 4 |
|   |   |   | 4 |
|   |   | 5 | 5 |
|   |   | 5 | 5 |
| 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 |

Illustration of the “append” problem:

```
(define a (list 1 2 3))
(define b (list 4 5 6))
(define c (append a b))
```

```
a
b
c
(set-car! a 0)
(set-car! b 0)
a
b
c
```

should out put:

Welcome to DrScheme, version 301.14-svn12may2006.

Language: Standard (R5RS).

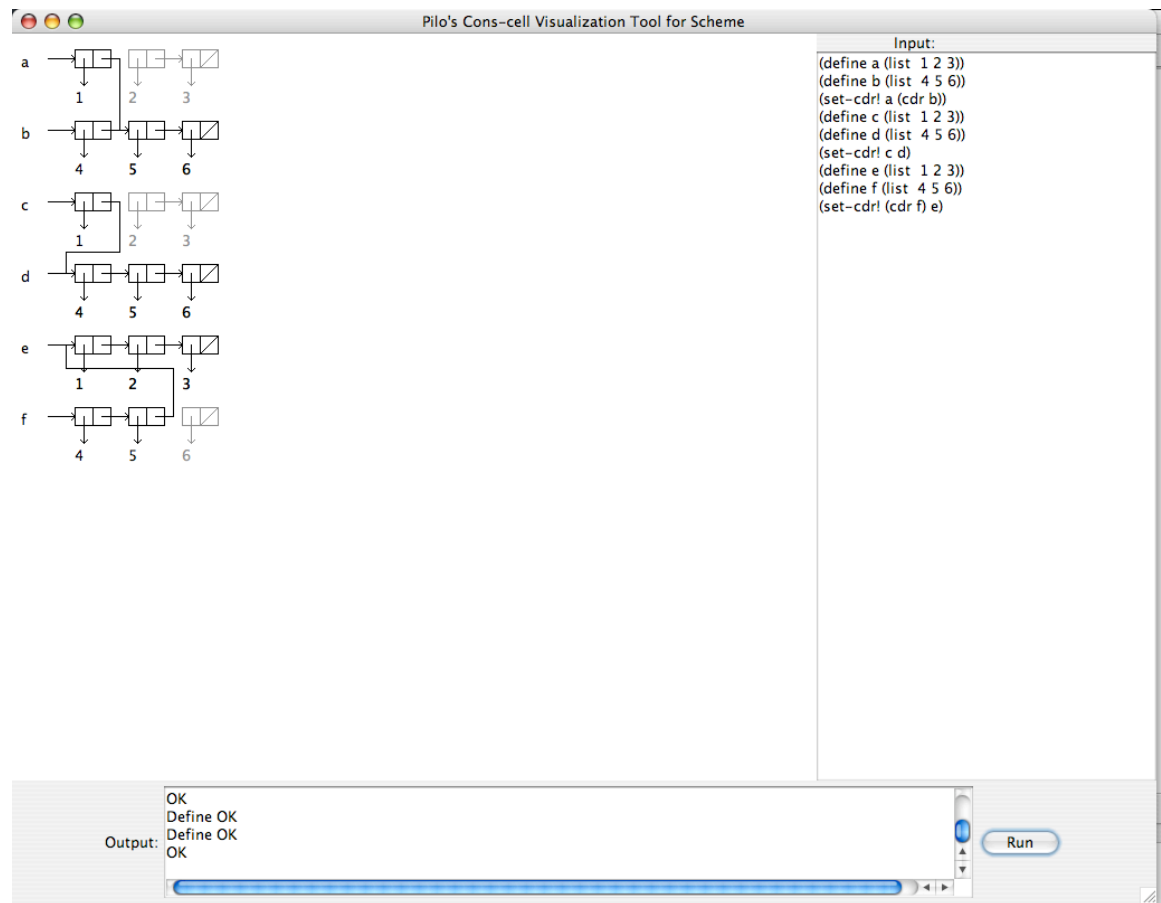
```
(1 2 3)
(4 5 6)
(1 2 3 4 5 6)
(0 2 3)
(0 5 6)
(1 2 3 0 5 6)
```

instead of :

```
PVTS>>
OK
OK
OK
(1 2 3)
(4 5 6)
(1 2 3 4 5 6)
```

OK  
 OK  
 (0 2 3)  
 (0 5 6)  
 (0 2 3 0 5 6)

Picture of the GUI as of **v8.1**:



Fooscape: website: <http://www.nickerson.to/visprog/CH2/progvis24.htm>

#### 2.4.4. Kaestle, FooScope

Kaestle (Boecker 1986) is in many ways similar to Incense. It is built on top of LISP instead of Mesa. Since the LISP cell structure is uniform, the system can automatically generate a visual representation for any data structure.

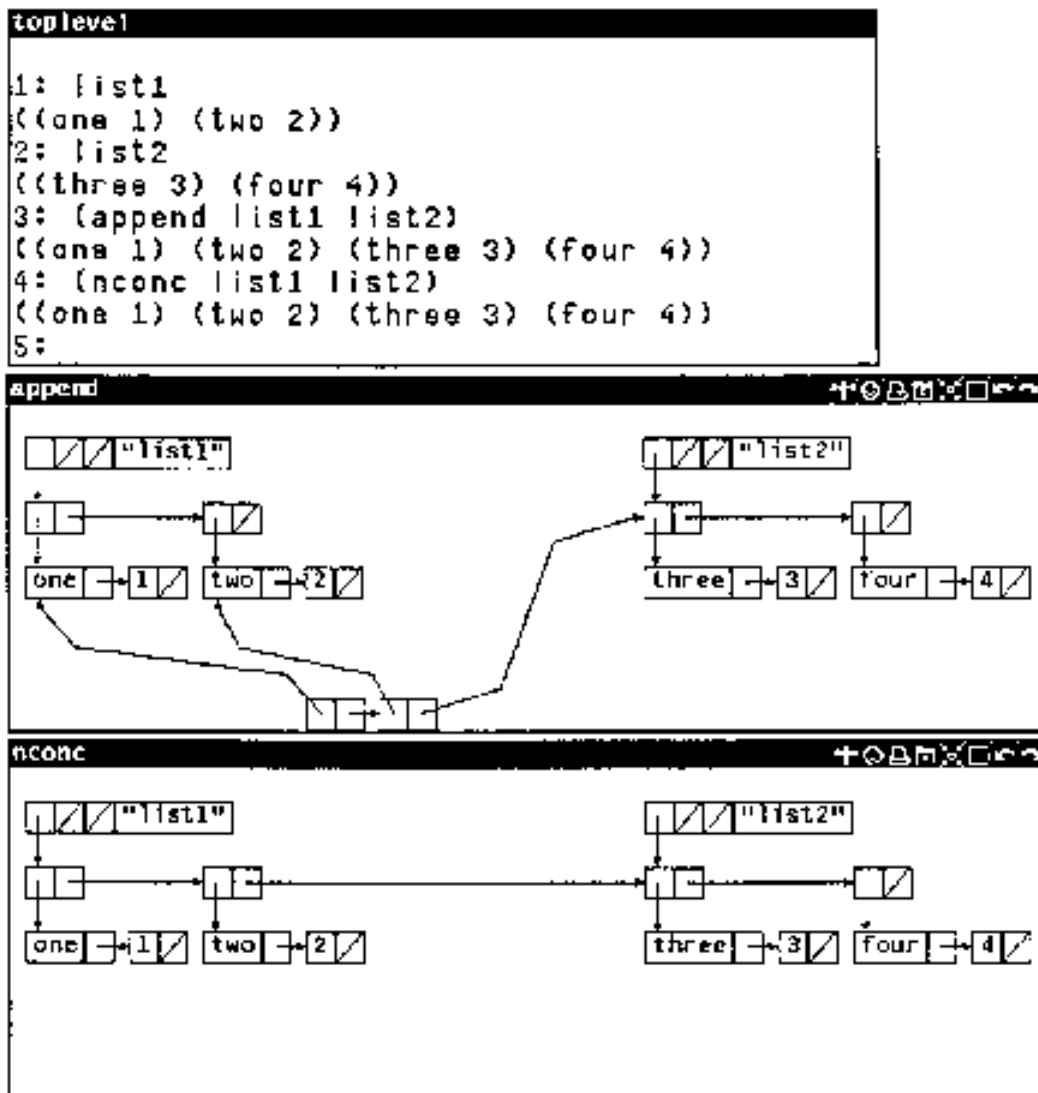


Figure 2.40. Castle. From Boecker (1986).

Fig

Fooscape, also mentioned in Boecker 1986, operates at the function level. It is described as a landscape of functions:

**Picture of the GUI as of version v18.5:**

### Concells Viewer

### Environment Viewer

| Lists Variables Table: |                     | Other Variables Table: |        | Procedures Table: |                    |
|------------------------|---------------------|------------------------|--------|-------------------|--------------------|
| Name:                  | Value:              | Name:                  | Value: | Name:             | Value:             |
| list4                  | (3 4 8 (1 2))       |                        |        | isin              | (cond ((null? l... |
| list3                  | ((3 4) 2 1 1 ((4... |                        |        | unionofwoesets    | (cond ((null? ...  |
| list2                  | (1 1 2 4 5)         |                        |        | teststruct        | (cond ((and (...   |
| list1                  | (1 2 3 4)           |                        |        | convertlistoset   | (cond ((null? l... |
| list0                  | 0                   |                        |        |                   |                    |

---

Function calls zoom:

Function Calls Options:

View modes:  Left aligned. Better for small trees.  Left aligned. Better for big trees.

Show only first character of function names.

### Pilo's Visual Tools for Scheme

File View Help

Input:

```
(or (and (list? lis1) (list? lis2)) (and (teststruct (car lis1) (car lis2)) (teststruct (cdr lis1) (cdr lis2))))
((and (list? lis1) (list? lis2)) (and (teststruct (car lis1) (car lis2)) (teststruct (cdr lis1) (cdr lis2))))
((and (not(list? lis1)) (not(list? lis2))) #t
)
)
)
(teststruct '(3 5 3) 9)
(teststruct '(1 2 3) '(3 2 1))
(teststruct '(1 (2 (2 3))) '(2(3) 2))
(teststruct '(1 (2 2) (3 (4 3))) '(4 (5 5 5) (3 (5 4))))
(teststruct '(1 (2 2) (3 (4 3))) '(4 (5 5 5) (3 (5 3 4))))
(teststruct '(1) (2 3))
```

"end of test"

Controls:

Viewers:

Output: