# *MANUAL CHANGE INFORMATION*

ARTIFICIAL INTELLIGENCE MACHINES DIVISION

**PRODUCT** 4404 SMALLTALK-80 INTRODUCTION USERS **PART NO** 070-5606-00

**PRODUCT GROUP** 07 **CHANGE NO** C15606 **DATE** DEC 1985

**This is an ADDITION package.**

1. Insert the attached addition, Smalltalk-80 Application Note (Subtask Support and Mangement) into the NOTES section of your manual.
2. Replace the title and manual revision pages at the front of your manual.
3. Keep this cover sheet in the CHANGE INFORMATION section at the back of this manual for a permanent record.

TEK USERS MANUAL

Part No. 070-5606-00
Product Group 07

# 4404

# ARTIFICIAL INTELLIGENCE SYSTEM

## INTRODUCTION TO THE SMALLTALK-80 SYSTEM

*Please Check at the Rear of this Manual for NOTES and CHANGE INFORMATION*

First Printing DEC 1984
Revised DEC 1985

Tektronix
COMMITTED TO EXCELLENCE

**PLEASE FORWARD ALL MAIL TO:**

**Artificial Intelligence Machines**
**Tektronix, Inc.**
**P.O. Box 1000 M.S. 60-405**
**Wilsonville, Oregon 97070**
**Attn: AIM Documentation**

# MANUAL REVISION STATUS

**PRODUCT: 4404 Artificial Intelligence System Smalltalk-80 System**

This manual supports the following versions of this product: Version T2.2.0

| REV DATE | DESCRIPTION |
|---|---|
| DEC 1984 | Original Issue |
| DEC 1985 | Subtask Management Notes |

# Application Note

# SUBTASK SUPPORT
## AND MANAGEMENT

Support for subtasks created by the operating system is available in the Smalltalk-80 system. Although Smalltalk has its own processes, Smalltalk can also create and communicate with subtasks created by the operating system. Smalltalk processes provide independent paths of control within Smalltalk. All Smalltalk processes share access to the same set of objects. Operating system subtasks provide access to other executable programs. These subtasks are useful for running OS utilities and commands, as well as communicating with applications and programs written in other languages. Subtasks can be executed without leaving the Smalltalk environment. Protocol supporting subtasks is found in the classes **Subtask**, **Pipe**, **PipeStream** and its subclasses. The objective of this support is to make the job of creating, running and communicating with the subtasks straightforward. Interfaces to OS signals, program parameters, environment variables, and subtask priorities are also supported.

## OVERVIEW OF SUBTASKS AND PIPES

The operating system on the 4404 supports multi-tasking. With Smalltalk's interface to the operating system via system calls, it is possible for Smalltalk to utilize this facility. Multi-tasking is achieved by spawning new tasks. A newly spawned task is called a child task or a subtask. The original task is referred to as the parent task. The child task is a "copy" of the parent task -- it shares memory and other resources with the parent task. Since only one task can execute at a time, cpu time is also shared, initially in a predetermined fashion. Common practice is for the spawned child task to perform some chore, and then report back to the parent task. After reporting, the child task disappears. This disappearance is known as the termination of the child task. A parent may chose to relinquish use of the CPU until a subtask terminates. It does this by an operation called waiting. While waiting, the parent task is blocked and cannot do anything else until the child task terminates.

The child task's chore is often accomplished by finding some other program to do the work. The use of this other program is known as an exec operation (for execute). In an exec operation, the original spawned task "turns itself into" the other program, so the other program becomes the child task.

Often times, a parent task may want to communicate with a child task. Information can be sent to and from the child task by using pipes, however, each pipe can send information in only one direction. If communication in two directions is desired, two pipes must be used. Pipes are similar to files with two critical differences.

- Files can be reopened many times. Pipes can only be opened once. Once a pipe is closed it is gone.
- Files can be reset and repositioned. It is not possible to reposition a pipe.

Usually the parent task creates a pipe. Each end of the pipe is assigned a file descriptor, one for reading and one for writing. When a subtask is created it inherits these open file descriptors. The parent task remembers one file descriptor, the one which is appropriate for its direction of communication. For example, if the parent task wants to send information to the child, the parent remembers the file descriptor for writing. Since the parent will not be using the reading end of the

pipe, it should close this unused end. The child task must also remember the appropriate file descriptor and close the file descriptor corresponding to the unused end of the pipe. Neglecting to close these unused pipe file descriptors might mean the task could run out of file descriptors, since there is a limit of 32 open file descriptors per task.

Sometimes it is not possible for the child task to know that it should use the pipe's file descriptors for reading and writing. For instance, the child task might want to exec a program that writes on standard output. Even if the program were aware of the use of pipes, it may not be possible for the program to modify itself to use the pipe's file descriptor for writing. In this case, it is possible for the child task to redirect its I/O by mapping its pipe descriptors to known file descriptors. (This mapping is accomplished through the use of the **dups** system call. See the *4404 Reference Manual* for more details.) Once a pipe's file descriptor is mapped, it becomes obsolete and should be closed. For example, the child task may want to write to the pipe, but the program is designed so write operations go to standard output. The write file descriptor of the pipe must be mapped to standard output's file descriptor (1), and the write file descriptor should be closed. The effect of the mapping in this example is for write operations on standard output from the child task to be performed on the write end of the pipe instead.

## CLASS DESCRIPTIONS

Instances of the class **Subtask** represent operating system child tasks. The process of creating an instance of this class includes specifying an executable program and, optionally, arguments to the program and environmental variables. A block representing code to be executed by the child task can also be specified. **Subtask** contains protocol for invoking, terminating, and waiting for a child task. The invocation of a child task may include modification of the child's priorities and environment variables. The metaclass contains protocol for managing these child tasks.

The class **Pipe** represents an operating system pipe. Instance creation causes a pipe to be created with two open file descriptors, one for each end of the pipe. Protocol exists for mapping ends of the pipe to arbitrary file descriptors.

The class **PipeStream**, a subclass of **ExternalStream**, is an abstract class. Since instances of **ExternalStream** are positionable and pipes are not positionable, one of **PipeStream's** purposes is to provide protocol for filtering out inappropriate inherited methods. It also contains a method for closing a **PipeStream** or one of its subclasses, which also closes its associated pipe file descriptor. Protocol also exists for mode changes to binary or text. **PipeStream** has two subclasses, **PipeReadStream** and **PipeWriteStream**. Each of these subclasses is created by opening on an instance of **Pipe**. Each has protocol appropriate for its function: **PipeReadStream** has protocol for streaming over data read from a pipe. **PipeReadStream** buffers its data from the pipe. However, **PipeWriteStream** does not buffer its data. Both of these classes support non-homogeneous accesses, that is, reading or writing different sized pieces of data. These classes inherit higher

level protocol involving:

- Padding
- Accessing strings, numbers and words
- **fileIn** and **fileOut**

## SUBTASK EXAMPLES

Here is an example which uses **Subtask**. Assume the existence of an executable file **/bin/simpleUtility**, a program with no input, output, or arguments.

```
executeSimple
      "Execute a pretend program."
      | task |
      task ← Subtask fork: '/bin/simpleUtility' then: [].
      task start.
      task waitOn.
      task release.
```

The method **fork:then:** creates an instance of **Subtask**. This object, assigned to the variable **task**, contains all the information needed to create an OS subtask to execute the program **/bin/simpleUtility**. However, an actual subtask is not created by this method. The method **start** issues the system calls **vfork** and **exec** to create and run the subtask. The method **waitOn** instructs the currently executing Smalltalk process to wait for the subtask to terminate. **Release** discards the **Subtask** object.

Here is a somewhat more complicated example:

```
execSystemUtility: aCommand
      | pipe task inputSide resultOfProgram |
      pipe← Pipe new.
      task ← Subtask fork: aCommand then: [
                  pipe mapWriteTo: 1.
                  pipe mapWriteTo: 2.
                  pipe closeWrite; closeRead].
      task start.
      pipe closeWrite.
      inputSide ← PipeReadStream openOn: pipe.
      resultOfProgram ← inputSide contentsOfEntireFile.

      task waitOn.
      inputSide close.
      task release.
      ↑resultOfProgram
```

In this method, **execSystemUtility:**, a pipe is created to establish one-way communication with the subtask. (Two pipes are required for two-way communication.) The code in the block is executed by the subtask after the vfork

system call and before the exec system call. All the rest of the code in **execSystemUtility:** is executed by the parent task.

Pipe connections in the child task are established in the block. In this case, the child task's standard output (file descriptor 1) and standard error (file descriptor 2) are redirected to the pipe through the use of the **mapWriteTo:** method. When the child task writes to standard output or standard error, this mapping causes the write operations to be directed to the write side of the pipe. Since the write end of the pipe has been redirected, it is a good idea to close the write end with the message **closeWrite**. In addition to closing redirected ends of the pipe in the child task, unused ends of the pipe should be closed in both the parent and child tasks. In this case, the pipe read end is unused in the subtask, and the pipe write end is unused in the parent task.

The net affect of all this closing and mapping is that the child task (whose code is executed in the block) closes the read side of the pipe because it is unused and closes the write side of the pipe because it has mapped the write side to standard output and standard error. The parent task closes its unused end of the pipe, which is the write side. The parent task also creates a Smalltalk object for reading from the pipe, an instance of **PipeReadStream** called **inputSide**. **InputSide** inherits protocol from **PipeStream** and consequently **ExternalStream**. Although other methods may be used to read from the pipe, here, the method **contentsOfEntireFile** is used to read all the data from the pipe, and the pipe is closed after use.

The following method, found in **execSystemUtility:withArgs: TekSystemCall class**, in addition to havinf the same functionality as the method immediately above, also has error checking and passes arguments to the executable program, **aCommand**.

```
execSystemUtility: aCommand withArgs: anOrderedCollection
      | pipe task inputSide resultOfProgram |
      pipe ← Pipe new.
      task ← Subtask
                  fork: aCommand
                  withArgs: anOrderedCollection
                  then:
                        [pipe mapWriteTo: 1.
                        pipe mapWriteTo: 2.
                        pipe closeWrite; closeRead].
      task start isNil
            ifTrue:
                  [pipe closeWrite; closeRead.
                  self error: 'Cannot execute ' , aCommand].
      pipe closeWrite.
      Cursor execute
            showWhile:
                  [inputSide ← PipeReadStream openOn: pipe.
                  resultOfProgram ← inputSide
                  contentsOfEntireFile].
      task waitOn.
      inputSide close.
      task abnormalTermination ifTrue:
            [self error: 'Error from system utility: ' ,
                  (resultOfProgram copyUpTo: Character cr)].
      task release.
      ↑resultOfProgram
```

This method contains code to

- Pass arguments to the program in the form of an **OrderedCollection**.
- Check for failure of the child task (**task start isNil**).
- Test for abnormal termination of the child task.

Failure of the child task necessitates the closing of any pipes created for use in the subtask. The parent task which creates these **Pipes** is responsible for closing them. Neglecting to close these pipes might mean the Smalltalk parent task could run out of file descriptors.

## *Examples Using* execSystemUtility:withArgs:

Here are some examples that demonstrate how to use **execSystemUtility:withArgs:**.

To execute a program with arguments:

```
fileName ← 'timingData'.
flags ← '+sa'.
TekSystemCall
      execSystemUtility: '/bin/dir'
      withArgs: (OrderedCollection with: fileName with: flags).
```

The next example executes a shell with a **+c** option. The **+c** option tells the shell to read the rest of the arguments as a command to itself. The effect is a directory listing with the shell providing wildcard expansion.

```
pattern ← '/smalltalk/de*'.
nameList ← TekSystemCall
            execSystemUtility: '/bin/shell'
            withArgs: (OrderedCollection
                        with: '+c'
                        with: '/bin/dir +s ' , pattern)
```

Besides taking advantage of the shell's wildcard expansions, you can also use aliases which are stored in the **.shellhistory** file. (See the *4404 Reference Manual* — the **shell** command for more details.)

```
TekSystemCall
      execSystemUtility: '/bin/shell'
      withArgs: (OrderedCollection with: '+c' with: 'df')
```

where **df** is an alias for **free /dev/disk**.

To execute a program with no arguments substitute an empty **OrderedCollection** for the second argument.

```
TekSystemCall
      execSystemUtility: '/bin/date'
      withArgs: OrderedCollection new.
```

## *Environment Variables*

The Smalltalk-80 system's interface to subtasks also supports environment variables. (See the *4404 Reference Manual* for more details.) In general, when a program is invoked, the operating system passes arguments and environment variables to the program. Standard environment variables include **HOME** — a home directory specification and **PATH** — a search path specification. Environments are a way to pass information by name. This can be viewed as setting a context for execution. Instances of subtask are created with a default environment, the environment with which Smalltalk was invoked. The method **Subtask class copyEnvironment** answers a copy of the default environment. This copy is in dictionary format for easy modification. The method **Subtask environment:** assigns an environment to the **Subtask** instance which passes it to the executed program. Here is an example of use of a modified environment which specifies that **/smalltalk** is the current **HOME** directory.

```
execProgram: aCommand
      | task env |
      task ← Subtask
                        fork: aCommand
                        then: [].
      env ← Subtask copyEnvironment.
      env at: #HOME put: '/smalltalk'.
      task environment: env.
      task start isNil
            ifTrue:
                  [self error: 'Cannot execute ' , aCommand].
      task waitOn.
      task abnormalTermination ifTrue: [self error: 'Error from ' , aCommand].
      task release.
```

## *Signals*

Operating system signals (colloquially referred to as *interrupts*) can be intercepted, ignored, or set to a default action by using protocol in **TekSystemCall** class. Usually, the default action upon receipt of an interrupt is task termination. (See the *4404 Refence Manual* – the **int** command and **cpint** system call for more details.) Sometimes it is desirable for the child task to intercept, or modify, its reaction to an interrupt. Protocol to modify these reactions can be added to the block which is an argument to the **Subtask** instance creation methods. In our next example, **ScreenController forkOSShell** , the method **fork:withArgs:then:** is passed a block which modifies some of these reactions. Code in this block is executed by the child task only. The reaction to several interrupts is modified with the method **setInterrupt:to:** in both the parent and child task. Here is a simplified and stripped down copy of the method **ScreenController forkOSShell**.

```
forkOSshell
      "Simplified for example."

      | location task oldSIGHUPValue oldSIGINTValue oldSIGQUITValue
            sysCall oldSIGTERMValue |
      oldSIGHUPValue ← TekSystemCall setInterrupt: 1 to: 1.
      oldSIGINTValue ← TekSystemCall setInterrupt: 2 to: 1.
      oldSIGQUITValue ← TekSystemCall setInterrupt: 3 to: 1.
      oldSIGTERMValue ← TekSystemCall setInterrupt: 11 to: 1.
      task ← Subtask fork: '/bin/shell' withArgs:
            (OrderedCollection with: '+i')
            then: [
            TekSystemCall setInterrupt: 1 to: 0.
            TekSystemCall setInterrupt: 2 to: 0.
            TekSystemCall setInterrupt: 3 to: 0.
            TekSystemCall setInterrupt: 11 to: 0.
            sysCall ← TekSystemCall terminalOn.
            sysCall value].
      error ← task start.
      error isNil
            ifFalse: [task absoluteWait.
                  task release].

      TekSystemCall setInterrupt: 1 to: oldSIGHUPValue.
      TekSystemCall setInterrupt: 2 to: oldSIGINTValue.
      TekSystemCall setInterrupt: 3 to: oldSIGQUITValue.
      TekSystemCall setInterrupt: 11 to: oldSIGTERMValue.
      sysCall ← TekSystemCall terminalOff.
      sysCall value.
      ScheduledControllers restore.
      error isNil ifTrue: [↑self error: 'Cannot fork shell']
```

Interrupt action is modified in both the parent and child tasks by using the method **setInterrupt:to:**, which returns the previous action for that interrupt. First, the parent interrupt actions are saved in temporary variables while setting interrupt action to 1, which means to ignore the interrupt. In the subtask, these same interrupts are reset to the default action, by making the interrupt action 0. After the subtask has completed, the interrupts in the parent task are set back to their original values. This subtask runs by using the protocol previously described, but the parent task waits for the child task to terminate by using the method **absoluteWait**. This method actually shuts down the Smalltalk parent task so it receives no time slice from the operating system scheduler. This strategy of waiting makes the subtask more efficient because the parent task cannot steal any processing power. However, Smalltalk cannot run until the child task has terminated. **AbsoluteWait** is not appropriate for any subtask that depends on the Smalltalk user interface.

## *Priorities and Two Way Communication*

Subtasks can be made to run more efficiently by changing their priorities. Sending the message **priority:** to an instance of **Subtask** modifies the invocation of a subtask so that it runs at the designated priority. Here is the definition of the method **Subtask priority:**.

```
priority: aPriority
      "Set the priority of the subtask.  Acceptable values range from 0 to
      25, zero being the highest and 25 being the lowest."

      aPriority < 0 | (aPriority > 25)
            ifTrue: [self error: 'Unacceptable priority value'].
      priority ← aPriority
```

The Smalltalk parent task initially has a priority of 10. If the child task is created with a higher priority than the parent task, it has a potential of taking control of the CPU. A higher priority task will not relinquish the CPU to a lower priority task unless the higher one is blocked or terminates. The next example increases the child task's priority and uses two pipes for two way communication. It is known that the child task in this case will be blocked while waiting for input, so the parent task will have a chance to run. The class **ShellInterface** creates a view that communicates with a shell (**/bin/script**) interactively. (Note that this class is not contained in the **standardImage** but defined in the file **/smalltalk/fileIn/Examples-Subtasking**.) After an instance of this class is created, it is initialized with the following method.

```
initialize
      | pipeIn pipeOut sysCall |
      command ← 'command'.
      result ← '' asText.
      pipeIn ← Pipe new.
      pipeOut ← Pipe new.
      shellTask ← Subtask fork: '/bin/script'
                        then:
                              [FileStream releaseStdRefs.
                              pipeIn mapWriteTo: 1.
                              pipeIn mapWriteTo: 2.
                              pipeIn closeWrite.
                              pipeIn closeRead.
                              pipeOut mapReadTo: 0.
                              pipeOut closeRead.
                              pipeOut closeWrite].
      shellTask enhancedPriority.
      shellTask start isNil
            ifTrue:
                  [pipeIn closeWrite; closeRead.
                  pipeOut closeWrite; closeRead.
                  self error: 'Cannot execute a shell'].

      pipeOut closeRead.
      pipeIn closeWrite.
      shellIn ← PipeReadStream openOn: pipeIn.
      shellOut ← PipeWriteStream openOn: pipeOut
```

This method uses two pipes for two way communication. It also assigns the highest possible priority to the subtask with the message **enhancedPriority**. When the view is closed, each instance of **ShellInterface** cleans up with the release method.

```
release
      | sysCall |
      "Close pipes for interactive shell"
      shellOut isValid ifFalse: [^self].
      shellIn close.
      shellOut close.
      shellTask kill.
      shellTask waitOn.
      shellTask release.

      TekSystemCall terminalOff value.
      TekSystemCall cursorOn value.
      Cursor cursorLink: true.
      Display enableCursorPanning.
      Display enableJoydiskPanning.
      Display setNormalVideo.
```

The shell subtask, **shellTask**, is terminated with the **kill** message. The parent task waits for the shell subtask to terminate and then releases the instance of **Subtask**. Other methods are invoked to reset states in case a subtask of the shell has modified the display.

## THE DETAILS OF STARTING A SUBTASK

Here is the **start** method taken from the class **Subtask**. This method contains low level details of how a subtask is actually exec'ed.

```
start
      "Start the receiver by executing a vfork, code to set up
       the child task (mainly communication and signal processing),
       and exec'ing the program.  If the exec fails terminate the
       child task.  The child task will inherit the priority of
       the Smalltalk task."

      | forker execer task |
      forker ← TekSystemCall vfork.
      environment isNil
            ifTrue: [execer ← TekSystemCall exec: program with: args]
            ifFalse: [execer ← TekSystemCall execve: program
               withArgs: args withEnv: environment].

      forker value.
      initBlock value.
      self priority notNil ifTrue: [(TekSystemCall setpr: priority) value].
      FileStream closeExternalReferences.
      execer invoke
            ifFalse:
                  [(TekSystemCall term: 0) value.
                  self taskId: nil.
                  ↑nil].
      self taskId: execer DOOut.
      ScheduledSubtasks add: self.
      self criticalSection: [self status: #running].
```

Subtasks are run from Smalltalk by making two essential system calls. The **vfork** system call, **forker**, creates a child task which shares memory with the parent task. The **exec/execve** system call, **execer**, transforms the child task so it is no longer running Smalltalk, but is executing the specified binary file, **program**. **Execer** is created with either the **exec** system call (**exec:with:**) or with the **execve** system call (**execve:withArgs:withEnv:**), depending on whether an environment is passed to the binary program.

Before the **exec/execve** invocation (**execer value**), communications and signals must be set up. When an instance of **Subtask** is created, potentially, a block is passed as an argument containing code for setting up communications and signals in the child task. This block was stored in the instance variable, **initBlock**, which is evaluated at this point. Priorities are also assigned at this time, and files belonging to the parent task Smalltalk are closed with the expression **FileStream closeExternalReferences**. After the child task has been spawned and control returns to the parent, the task identification number is recorded by the parent task (**self taskId: execer DOOut**), the child task is added to the list of managed subtasks (**ScheduledSubtasks add: self**), and its status is recorded. If the **execer**

call fails, the child task terminates itself with the **term:** method and the parent task returns **nil**.

At the **vfork** invocation (**forker value**), control of the Smalltalk virtual machine transfers to the child task. The child task continues executing with no access to the keyboard or the mouse until the **execer** call is invoked. This means that the code from **forker value** to **execer invoke** is only executed by the child task. In addition,

```
(TekSystemCall term: 0) value.
```

is executed by the child task to terminate itself if the **execer** call fails. Then the parent task resumes control and executes the statements

```
self taskId: nil.
^nil
```

to indicate failure of the **execer** call. The next statements, starting with **self taskId: execer DOOut**, are also executed by the parent task, but only if the **execer** call was successful. Whenever the child task is blocked or the child task terminates, control reverts back to the parent task. The parent task initially resumes control in the state left by the child task before the **forker** invocation.

# *MANUAL CHANGE INFORMATION*

ARTIFICIAL INTELLIGENCE MACHINES DIVISION

**PRODUCT** 4404 SMALLTALK-80 INTRODUCTION USERS **PART NO** 070-5606-00

**PRODUCT GROUP** 07 **CHANGE NO** C25606 **DATE** APR 1986

**This is an ADDITION package.**

1. Insert the attached Smalltalk-80 LOS User Notes into the NOTES section of your manual.
2. Replace the title and manual revision pages at the front of your manual.
3. Keep this cover sheet in the CHANGE INFORMATION section at the back of this manual for a permanent record.

TEK USERS MANUAL

Part No. 070-5606-00
Product Group 07

# 4404

# ARTIFICIAL INTELLIGENCE SYSTEM

## INTRODUCTION TO THE SMALLTALK-80 SYSTEM

*Please Check at the Rear of this Manual for NOTES and CHANGE INFORMATION*

First Printing DEC 1984
Revised APR 1986

Tektronix
COMMITTED TO EXCELLENCE

**PLEASE FORWARD ALL MAIL TO:**

**Artificial Intelligence Machines**
**Tektronix, Inc.**
**P.O. Box 1000 M.S. 60-405**
**Wilsonville, Oregon 97070**
**Attn: AIM Documentation**

# MANUAL REVISION STATUS

**PRODUCT: 4405 ARTIFICIAL INTELLIGENCE SYSTEM SMALLTALK-80 SYSTEM**

This manual supports the following versions of this product: Version T2.2.0

| REV DATE | DESCRIPTION |
|---|---|
| DEC 1984 | Original Issue |
| DEC 1985 | Subtask Management Notes |
| APR 1986 | LOS Users Notes |

# Table of Contents

# Figures



# Tables

# Section 1

# Smalltalk-80 LOS User Notes

These notes document the significant differences between the Small Object Space (SOS) Smalltalk-80[1] system and the Large Object Space (LOS) Smalltalk-80 system Version T2.2.0 designed for the Tektronix 4405 and 4406 Artificial Intelligence Machines.

The version of the Smalltalk-80 system that runs on the 4404 Artificial Intelligence System (AIS) is limited to the creation of about 32,000 objects, that is, its Smalltalk interpreter – the Small Object Space (SOS) interpreter – implements object-oriented pointers (oops) as 16-bit words. This limitation has been removed in the new interpreter – the Large Object Space (LOS) interpreter – written for the 4405 and 4406 AIS machines.

## ABOUT THESE NOTES

This document consists of the notes themselves and three appendices. The contents are:

- The notes themselves document the differences between the SOS and LOS Smalltalk systems. Also, significant modifications and enhancements to the LOS image are described.
- Appendix A *Conversion of SOS to LOS Images.* This tells you how to transfer work you have done in a SOS image to a LOS image. This enables you to build upon work you have already done and also to take advantage of the LOS system performance and functional enhancements.
- Appendix B *Smalltalk-80 Version T2.2.0 Files.* For your convenience, you will find a list of all files associated with the LOS Version T2.2.0 release.
- Appendix C *Changes in the Smalltalk-80 Images.* For your convenience, you will find a list of all changes to classes and methods in the new releases of the SOS and LOS images.

## The Blue Book Specification

The de facto specification for SOS Smalltalk-80 is the Addison-Wesley Smalltalk-80 book by Goldberg and Robson. (This book is sometimes colloquially referred to as the "blue book" in the Smalltalk-80 programming community.) References are made in these notes to specific chapters, sections, and pages in the Goldberg and Robson book:

- Goldberg, Adele and David Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, 1983.

The Goldberg and Robson book is a formal explanation and description of the Smalltalk-80 language. This includes not only the syntax of the language but also the classes of objects that make up the Smalltalk-80 virtual image. In Part Four, this book contains a detailed discussion of the implementation of the SOS virtual machine. (That is, Part Four is a *specification* of the

1. Smalltalk-80 is a Trademark of Xerox Corporation.

interpreter and object memory of a Small Object Space Smalltalk-80 system.)

# INVOKING SMALLTALK

The standard 4405 and 4406 AIM systems have resident on them two Smalltalk systems, each consisting of an interpreter and a virtual image: the SOS (Small Object Space) Smalltalk-80 system and the LOS (Large Object Space) Smalltalk-80 system. If you type:

*smalltalk*

at the system prompt, this brings up the LOS interpreter. (A special case exists here: if you have a 4405 machine *with just 1 megabyte of memory*, then typing *smalltalk* brings up the *SOS* interpreter and image. However, if you have *more than 1 megabyte of memory* in your 4405, then typing *smalltalk* brings up the LOS interpreter and image just as it does on a 4406 system.)

Typing

*smalltalk myImage*

brings up the LOS interpreter, if *myImage* is an LOS image or the SOS interpreter if *myImage* is an SOS image.

Also, suppose for some reason you want to bring up the SOS interpreter on the standard SOS image, then you can type:

*smalltalk +s*

Finally, if you have a 4405 system with just 1 megabyte of memory and you want to bring up the LOS image, you can type:

*smalltalk +l*

# THE LOS INTERPRETER

The LOS (Large Object Space) interpreter differs in a number of ways from the SOS (Small Object Space) interpreter developed for the 4404 AIM system. The principal differences are:

- The LOS interpreter has no inherent limit to the number of objects in an image. The SOS interpreter, on the other hand, supplies at most 32,767 objects in a single image.
- The size of objects is not limited to 128K bytes as in the SOS interpreter.
- The LOS interpreter uses 32-bit object-oriented pointers (oops) instead of 16-bit object-oriented pointers.
- SmallInteger values are in the range $-2^{30}$ to $2^{30}-1$.
- The LOS interpreter uses MC68020 microprocessor-specific instructions, which means that the LOS interpreter runs only on the 4405 and 4406 AIM systems.

# Object-Oriented Pointers

Oops (object-oriented pointers) are the values used by the interpreter to name objects. Of the 32 bits in an oop, only 29 are used to actually name objects. Thus, there is a theoretical maximum of about 500 million object in the system. A practical limit for the maximum number of objects on the 4405 and 4406 systems with the LOS interpreter depends on the average size of objects and your system memory configuration. With an average object size of 50 bytes, together with a practical object memory size of 6 to 10 megabytes, the system allows approximately 120,000 to 200,000 objects.

## Small Integers

In the SOS system, which uses a 15-bit integer representation, operations on 16- to 32-bit positive integers use special LargePositiveInteger primitives that are not as fast as the SmallInteger primitive operations. Thirty two bit and larger operations are forced to use code written in Smalltalk.

SmallIntegers in the LOS system are represented with 31 bits. Operations on 16 to 31 bit integers use SmallInteger primitive operations which run much faster than SOS LargePositiveInteger primitive operations. The LargePositiveInteger primitives are not used since SmallIntegers encompass most of the range (16 to 31 bits) in which these primitives worked. Operations on integers larger than 31 bits still use code written in Smalltalk.

Note that Table 1-1, *Smalltalk Implementation Characteristics*, shows that byte indexable and word indexable elements in the SOS and LOS implementations are the same size, whereas oop size and SmallInteger size are not. The *size* of byte indexable, word indexable, and object indexable objects is greatly enlarged in the LOS interpreter.

**Table 1-1**
*Smalltalk Implementation Characteristics*

| Characteristic | SOS (16-Bit) Interpreter | LOS (32-Bit) Interpreter |
|---|---|---|
| SmallInteger Size | −16384 to 16383 | $-2^{30}$ to $2^{30}-1$ |
| Maximum Number of Objects | 32767 | Memory Size Limited |
| Size of Byte Indexable Elements | 8 bits | 8 bits |
| Size of Word Indexable Elements | 16 bits | 16 bits |
| Size of Object Indexable Elements | 16 bits | 32 bits |
| Maximum Size of Byte Indexable Objects | 128 K Elements | Memory Size Limited |
| Maximum Size of Word Indexable Objects | 64 K Elements | Memory Size Limited |
| Maximum Size of Object Indexable Objects | 64 K Elements | Memory Size Limited |

# LOS Interpreter Design Characteristics

The LOS interpreter has some important characteristics relevant to its overall design. In the LOS system, object management has been considerably changed from the SOS specification – there is no object table any longer. The class CompiledMethod has been substantially changed from that specified in Goldberg and Robson. CompiledMethod now looks and behaves much more like other classes. MethodDictionaries also have a different structure.

## The Object Table

In Goldberg and Robson in the chapter *Formal Specification of Object Memory*, the authors specify the structure of the object table for the SOS system. The SOS interpreter implements this 32K entry table, in which the maximum number of objects is limited to the number of entries in the object table. The LOS interpreter does not use an object table, since, with oops of 32 bits, the number of possible entries in such an object table would be impractical to manage. The benefits of eliminating an object table are:

- There is no *object-table* limit to the number of objects in the system.
- There is no extra level of indirection involved with every single operation on objects – creation, destruction, and manipulation.

There is, however, a penalty incurred with several rarely used methods, such as, become:. See *Primitive Methods* for a discussion of this.

## CompiledMethods

Class CompiledMethod has changed to be more consistent with the standard Smalltalk object structure. See Figure 1-1, *LOS Structure of an Instance of CompiledMethod.* Instances of CompiledMethod now consist of three objects: the CompiledMethod structure itself, an instance of LiteralArray, and an instance of ByteCodeArray. Note that the *Source Code Reference* field in the CompiledMethod structure contains a reference to the source code on disk. The SOS implementation folded the source code reference into the bytecodes.

This new representation permits the creation of subclasses of CompiledMethod. Protocol for CompiledMethod now formally supports access to the source code reference. The source code reference in instances of CompiledMethod is divided into two fields. The three high order bits represent a zero-based reference into the global variable SourceFiles. This global contains an array of files. The remaining 27 low order bits in this SmallInteger represent the position of this method's source code in the file referenced by the three high order bits.

The new representation eliminates the need for special primitives for creating (newMethod:header:) and accessing (literalAt: and literalAt:put:) instances of CompiledMethod.

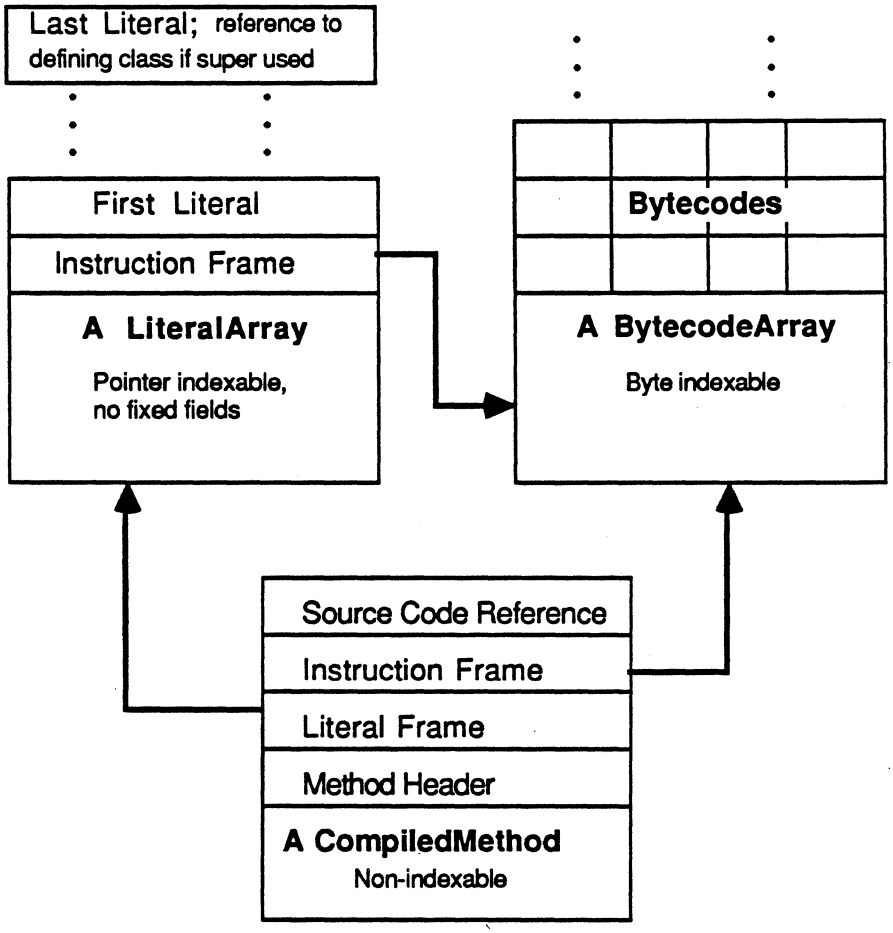


**Figure 1-1. LOS Structure of an Instance of CompiledMethod.**

## Method Dictionaries

The representation of instances of MethodDictionary has changed. See Figure 1-2, *Structures of Instances of MethodDictionary*. In the SOS interpreter, the instances of method dictionaries contain the keys (CompiledMethod selector names) as part of the method dictionary object itself. This has been changed in the LOS interpreter. The keys, instead of being part of the MethodDictionary object itself now are contained in a separate Array object that holds the selector names. This redesign was motivated primarily by the wish to eliminate the use of become:. Previously, become: was used to accomplish an atomic update operation. Since become: is now a relatively slow operation, the atomic update is now accomplished by methods that rely on the separation of keys.

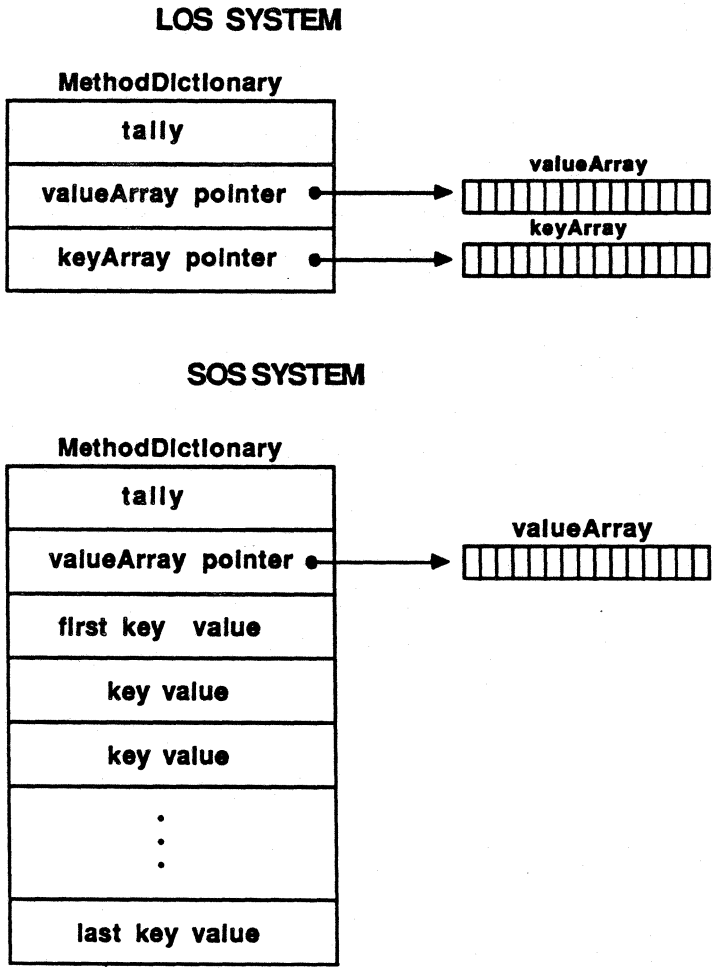


**Figure 1-2. Structures of Instances of MethodDictionary.**

# Primitive Methods

Some Smalltalk methods are implemented by making machine language calls directly. These methods are called primitive methods. In the LOS implementaion, some of the primitive methods specified in the Goldberg and Robson book have been eliminated, others have been added to the system, and others have changed function.

## Unimplemented Primitives

In the Goldberg and Robson book, the chapter *Formal Specification of the Primitive Methods* gives a list of primitive methods along with their associated primitive indexes. A number of these are not appplicable to the LOS system and, hence, are not implemented.

Primitives 21 through 37 are defined to perform arithmetic on 16-bit LargePositiveInteger objects. In the LOS interpreter, primitive indexes 21 through 37 have been eliminated since the LOS interpreter uses 31-bit wide small integer values. There is no meaningful distinction for most practical applications between the instances of SmallInteger and the instances of LargePositiveInteger and LargeNegativeInteger falling within the 31-bit limit. Thirty bit integer arithmetic operations are done at the machine language level. (Arbitrary precision arithmetical operations can still be done using Smalltalk methods for integers that cannot be expressed in 31 bits.)

Primitive indexes 68, 69, and 79, which deal with CompiledMethod have been eliminated due to the redesign of CompiledMethod.

Primitive index 76 (asObject) has been eliminated in the LOS interpreter because there is no object table. In the SOS interpreter, asOop and asObject functioned as inverses. This value would typically be used as a hash code for the object. Sending asOop to an object returns an integer representing its oop – the object table reference to the object. Sending asObject to an oop (represented by a SmallInteger) returns the object. Since there was a one-to-one correspondence between objects and object table reference values, you could be certain that two objects were the same object if they had the same asOop value in the SOS system. However, since the LOS interpreter does not use an object table, the asOop method returns a value with a different meaning. This value is the hash value calculated for each object at its creation. These values are not guaranteed to be unique for every object. In practice, the vast majority of hash values are unique, but since there is no longer a guaranteed unique asOop value for each object, the rationale for asObject is gone. Thus, asObject has been eliminated.

Primitive index 78 (nextInstance) has been eliminated since there is no inherent ordering of objects in this implementation. nextInstance was typically used in conjunction with someInstance to obtain all existing instances of some class. In the LOS interpreter, allInstances has been implemented as a new primitive method.

## New Primitives

The new primitives added to the LOS interpreter fall into four categories:

- Object management – 3 primitives
- Floating point – 6 primitives

- String comparison – 1 primitive
- Instance Creation – 2 primitives

### Object Management Primitives

137 Smalltalk garbageCollect: – Force a garbage collection. The argument identifies a storage grade. The virtual image is partioned into grades containing objects of corresponding ages. That is, newer objects are contained in lower grades and older objects are contained in upper grades. Valid numbers for grades are 0-7 inclusive.

138 Smalltalk core – Answer an Array containing the number of objects in the system and the number of words they occupy. Note that the count may include garbage objects which are eligible for garbage collection.

143 Behavior allInstances – Answer an array containing of all instances of this class. This may include instances that are elibible for garbage collection.

### Floating Point Primitives

155 Float arcCos – Answers arccosine x, where x is the receiver.

154 Float arcSin – Answers arcsine x, where x is the receiver.

156 Float arcTan – Answers arctangent x, where x is the receiver.

157 Float exp – Answers $e^x$, where x is the receiver.

158 Float ln – Answers ln x, where x is the receiver.

159 Float log – Answers log x, where x is the receiver.

### String Comparison Primitive

148 string = – Answers true if the receiver and argument contain the same ASCII characters. Answers false if not. Fails if the class of the argument is different from the class of the receiver.

### Instance Creation Primitives

140 DisplayBitmap basicNew: and new: – Answer a new instance of DisplayBitmap with the number of indexable variables specified by the argument, anInteger.

141 ContextPart basicNew: and new: – Answer a new instance of the receiver with the number of indexable variables specified by the argument, anInteger. Use of this instantiation primitive enables the creation of subclasses MethodContext and BlockContext.

## Old Primitives That Function Differently

41–50 Float + through Float truncated – If the argument is a SmallInteger, it is converted to a Float number and there is no failure.

72 Object become: – This method is potentially "expensive" in the sense that in the LOS system it takes a relatively long time to execute. The primary reason for this is that the SOS interpreter relied on swapping object table references where as the LOS interpreter must actually manipulate oops in memory (since there is no object table in the LOS interpreter). Many special cases are optimized to minimize execution time but in the most general case, this primitive involves examining all objects in the virtual image. In the LOS system, you may want to find alternative ways to code algorithms that in the SOS system used become:.

75 Object asOop, Object hash, Symbol hash – In the SOS system, an object's object table index (returned by asOop) is frequently used as a hash value. In the LOS system, each object is assigned a hash value at creation. This is a 16-bit value. asOop is now defined to return this value. In a SOS system, there is a one-to-one correspondence between objects and asOop values. Since this is not true in the LOS system, asOop is no longer an invertible function as it was in a SOS system. See the earlier discussion of asObject.

112 SystemDictionary coreLeft – This returns an *estimate* of the amount of memory available for new objects. (Use primitive 138 instead.)

115 SystemDictionary oopsLeft – This returns an *estimate* of the number of oops remaining to be allocated based on the core left value divided by the average object size. (Use primitive 138 instead.)

116 SystemDictionary signal:atOopsLeft:wordsLeft: – Since oops and memory are not practical system limits, this functions as a no-op.

# IMAGE MODIFICATIONS

The LOS image has a large number of modifications (see Appendix C for a complete list). Here are some highlights.

- The classes ByteCodeArray and LiteralArray have been added to support the new definition of CompiledMethod. Classes Debugger, Compiler, and related classes have modifications pertaining to the new CompiledMethod.
- Class Float has some additional class variables to support exceptional floating point values (see *IEEE Floating Point Numbers*). New protocol has been added to deal with these values also.
- Many View subclasses (ListView, StandardSystemView, StringHolderView, and TextView, for example) have been redefined for augmented access to fonts. PopUpMenu, StrikeFont, and TextStyle classes also changed and two new classes, StrikeFontManager and TextStyleManager, have been added. See *Enhanced Font Support* for more details.
- The SystemTracer class has been redefined and modified to produce a clone image in the LOS format.
- Additional protocol has been added to TekSystemCall related to pseudo-ttys and access to the machine name.

- Behavior, Class, and ClassDescription have modifications related to lazy mutation. See *Lazy Mutation* for a description of the differences apparent to users.

# User Interface Changes

## New Window Framing

Windows now frame by letting you switch between moving the top-left and bottom-right corners until you get them placed exactly how you want them.

When you want to frame a window via the normal user interface, for example, the "top-left" cursor appears, which you may move around on the screen. When it is approximately in the right position, you press the left mouse button, causing the "bottom-right" cursor to appear. Once you have located the bottom-right corner, you have two options. The first is to remove your finger from the mouse button completely; this has the effect of selecting the rectangle just framed. The second option is for you to lift your finger from the mouse for just an instant and to immediately press it again. This has the effect of moving the cursor back to the top-left corner of the rectangle, allowing you to adjust your original placement of that corner. When you are finished with the top-left corner, you again may move back to the bottom-right corner in the same manner, etc.

The determination of whether you have "quick-clicked" or not is made by an instance of class Delay, which is created in the method getFrame. There is a constant in this routine that specifies the time in milliseconds to wait. This constant is currently set at 250 (or 1/4 of a second); you can set it to another value by modifying the StandardSystemView getframe method.

## Blue Button Menu

The right button menu of StandardSystemViews has a new item – style. This allows a change of text style for a particular window, including its subviews. Available text styles are determined by the contents of StyleManager. See the System Workspace for an example of how to add text styles to your image.

# Enhanced Font Support

Smalltalk has new default fonts, a larger variety of fonts, and augmented access to the fonts. Available fonts range from very small to very large, serif and sans serif, and proportional and monospaced fonts. These fonts have an additional face – bold italic. Protocol for adding fonts and text styles to an image has been defined.

## Fonts

Three properties (family, face, and size) are commonly associated with a font. Family is the intrinsic property. Families are named and frequently protected by copyright. Examples include "Helvetica" and "Times Roman". Face is the emphatic property. Examples include Basal (no emphasis), Bold, Italic, BoldItalic, and Underlined. Size is the dimensional property. It is typically specified by the height of capital "A" in points (72nds of one inch), although such a measure is more meaningful on paper than on a display.

A Smalltalk-80 font is an instance of class StrikeFont, which represents a single combination of family, face, and size values with a bitmap for each character. In some cases, a face (other than Basal) is synthesized by bitmap manipulation of the Basal face. Examples include copying and offsetting (Bold), shearing (Italic), and underlining.

## Available Fonts

This product release includes the families: Pellucida[2] Sans-Serif, Pellucida Serif, Pellucida Typewriter, Xerox Sans-Serif, and Xerox Serif. The Pellucida Sans-Serif, Pellucida Serif, and Xerox families are proportionally spaced (individual characters within the same font have varying widths); the Pellucida Typewriter family is monospaced (individual characters within the same font have the same width). The Pellucida families are new to this product release; the Xerox families are the standard Smalltalk-80 Version 2 fonts.

Fonts are stored using a standard file format within the directory */fonts*. The name of a file in this directory should be the name of the font it holds suffixed with *.font*.

The name of a StrikeFont is a String with three components (family, size, and face) and no embedded spaces. The family component is the family name with spaces removed; the size component is the printString of the numeric size; and the face component is a String of length zero, one, or two encoding the emphasis. The supported face codes are "" (Basal), "B" (Bold), "I" (Italic), "X" (BoldItalic), "U" (Basal Underlined), "BU" (Bold Underlined), "IU" (Italic Underlined), and "XU" (BoldItalic Underlined). Examples of names include "PellucidaSans-Serif8", "XeroxSerif12I", and "Typewriter18BU".

The Pellucida Sans-Serif and Serif fonts are available in four non-synthetic faces (Basal, Bold, Italic, and BoldItalic) and seven sizes (8, 10, 12, 14, 18, 24, and 36 point); see Figure 1-3, *Tektronix Proportional Fonts (PellucidaSerif and PelluciaSans-Serif)*, for the character set ordering. The Pellucida Typewriter fonts are available in two non-synthetic faces (Basal and Bold) and four sizes (10, 12, 16, and 18 point); see Figure 1-4, *Tektronix Monospaced Fonts (Pellucia Typewriter) Part 1* and Figure 1-5, *Tektronix Monospace Fonts (Pellucida Typewriter) Part 2*, for the character set ordering. The Xerox fonts are available in three non-synthetic faces (Basal, Bold, and Italic) and two sizes (10 and 12 point), although the Sans-Serif Italic 10 point font is synthetic.

2. Pellucida is a registered trademark of Bigelow and Holmes.

## Interpreting Font Tables

A few notes on interpreting the font tables will be helpful in constructing an application. The spaces in the table that are blank do not have a printing character for the corresponding character code. The characters for ASCII 32 through ASCII 127 are present in both the monospaced and proportional fonts. The proportional fonts contain additional characters in ASCII 1 through ASCII 31. Many of these characters are compatible with those originally supplied by Xerox in the standard Smalltalk-80 Version 2 image.

"m space" is a blank character which is the height and width of the letter m. "n space" is a blank character which is the height and width of the letter n. "em" and "en" are dashes the width of the character "m" and "n", respectively.

## Reading and Writing

Smalltalk StrikeFont class has methods for reading and writing Tektronix font files. Note that whenever Smalltalk reads a Tektronix font file, it switches the character position of the uparrow character (↑) and left arrow (←) with the caret (^) and underscore (_) characters. Thus, if you ask, for instance, the character ↑ what its asciiValue is, you get 94.

The method to write a StrikeFont takes care to switch the positions of the ↑, ←, ^, and _ characters if the type of the strike font is either 1 (Tektronix monospaced) or 2 (Tektronix proportionally spaced). This ensures that the proportional or monospaced fonts written by Smalltalk have consistent character ordering.

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $10_0$ | $10_1$ | $10_{10}$ | $10_{11}$ | $11_{00}$ | $11_{01}$ | $11_{10}$ | $11_{11}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | ~ | space | 0 | @ | P | ‘ | p |
| 0 0 0 1 | ˇ | ffi | ! | 1 | A | Q | a | q |
| 0 0 1 0 | ¿ | ffl | ” | 2 | B | R | b | r |
| 0 0 1 1 | Ç | em | # | 3 | C | S | c | s |
| 0 1 0 0 | ¨ | fi | $ | 4 | D | T | d | t |
| 0 1 0 1 | ` | fl | % | 5 | E | U | e | u |
| 0 1 1 0 | ff | en | & | 6 | F | V | f | v |
| 0 1 1 1 | ‘ | ˘ | ’ | 7 | G | W | g | w |
| 1 0 0 0 | ¡ | — | ( | 8 | H | X | h | x |
| 1 0 0 1 | | n space | ) | 9 | I | Y | i | y |
| 1 0 1 0 | | ∞ | * | : | J | Z | j | z |
| 1 0 1 1 | ´ | ↑ | + | ; | K | [ | k | { |
| 1 1 0 0 | | ← | , | < | L | \ | \| | \| |
| 1 1 0 1 | | . | - | = | M | ] | m | } |
| 1 1 1 0 | ¯ | ~ | . | > | N | ^ | n | ~ |
| 1 1 1 1 | m space | ° | / | ? | O | _ | o | ■ |

**Figure 1-3. Tektronix Proportional Fonts (PellucidaSerif and PellucidaSans-Serif).**

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $10_0$ | $10_1$ | $10_{10}$ | $10_{11}$ | $11_{00}$ | $11_{01}$ | $11_{10}$ | $11_{11}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | | space | 0 | @ | P | ‘ | p |
| 0 0 0 1 | | | ! | 1 | A | Q | a | q |
| 0 0 1 0 | | | ” | 2 | B | R | b | r |
| 0 0 1 1 | | | # | 3 | C | S | c | s |
| 0 1 0 0 | | | $ | 4 | D | T | d | t |
| 0 1 0 1 | | | % | 5 | E | U | e | u |
| 0 1 1 0 | | | & | 6 | F | V | f | v |
| 0 1 1 1 | | | ’ | 7 | G | W | g | w |
| 1 0 0 0 | | | ( | 8 | H | X | h | x |
| 1 0 0 1 | | n space | ) | 9 | I | Y | i | y |
| 1 0 1 0 | | ↑ | * | : | J | Z | j | z |
| 1 0 1 1 | | ← | + | ; | K | [ | k | { |
| 1 1 0 0 | | | , | < | L | \ | \| | \| |
| 1 1 0 1 | | | - | = | M | ] | m | } |
| 1 1 1 0 | | | . | > | N | ^ | n | ~ |
| 1 1 1 1 | m space | | / | ? | O | _ | o | ■ |

**Figure 1-4. Tektronix Monospaced Fonts (Pellucida Typewriter) Part 1.**

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $10_{0_0}$ | $10_{0_1}$ | $10_{1_0}$ | $10_{1_1}$ | $11_{0_0}$ | $11_{0_1}$ | $11_{1_0}$ | $11_{1_1}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | NU 128 | DL 144 | SP 160 | 0 176 | – 192 | Ñ 208 | ◆ 224 | 240 |
| 0 0 0 1 | SH 129 | D1 145 | Ä 161 | 1 177 | ¢ 193 | ñ 209 | ■ 225 | 241 |
| 0 0 1 0 | SX 130 | D2 146 | ä 162 | 2 178 | ¦ 194 | ¿ 210 | HT 226 | 242 |
| 0 0 1 1 | EX 131 | D3 147 | Å 163 | 3 179 | † 195 | ¡ 211 | FF 227 | 243 |
| 0 1 0 0 | ET 132 | DA 148 | å 164 | 4 180 | □ 196 | α 212 | CR 228 | 244 |
| 0 1 0 1 | EQ 133 | NK 149 | Æ 165 | 5 181 | ■ 197 | σ 213 | LF 229 | 245 |
| 0 1 1 0 | AK 134 | SY 150 | æ 166 | 6 182 | ● 198 | τ 214 | o 230 | 246 |
| 0 1 1 1 | BL 135 | EB 151 | à 167 | 7 183 | Δ 199 | ρ 215 | ± 231 | 247 |
| 1 0 0 0 | BS 136 | CN 152 | Ç 168 | 8 184 | δ 200 | μ 216 | NL 232 | 248 |
| 1 0 0 1 | HT 137 | EM 153 | é 169 | 9 185 | λ 201 | Σ 217 | VT 233 | ≤ 249 |
| 1 0 1 0 | LF 138 | SB 154 | è 170 | ù 186 | | Ω 218 | 234 | ≥ 250 |
| 1 0 1 1 | VT 139 | EC 155 | Ö 171 | β 187 | | ∫ 219 | 235 | π 251 |
| 1 1 0 0 | FF 140 | FS 156 | ö 172 | ⊙ 188 | | 220 | 236 | ≠ 252 |
| 1 1 0 1 | CR 141 | GS 157 | ø 173 | ¤ 189 | | ÷ 221 | 237 | £ 253 |
| 1 1 1 0 | SO 142 | RS 158 | Ü 174 | § 190 | ¬ 206 | ≈ 222 | 238 | ■ 254 |
| 1 1 1 1 | SI 143 | US 159 | ü 175 | ▪▪ 191 | ∝ 207 | ⌐ 223 | 239 | DT 255 |

**Figure 1-5. Tektronix Monospaced Fonts (Pellucida Typewriter) Part 2.**

The new class StrikeFontManager is a subclass of Dictionary and stores Associations between String names and StrikeFonts. A single instance of StrikeFontManager is known as the global FontManager. Particularly useful messages to this object include:

```
FontManager inspect.
FontManager fontNames: anArray.
```

The inspect method opens a DictionaryInspector on FontManager.

The fontNames: method returns an Array of StrikeFonts corresponding to anArray of String names. It attempts to load a font from the system font directory (*/fonts*) if that font is not already resident (contained within FontManager). The name of a file in this directory should be the name of the font it holds suffixed with ".font". The method further attempts to synthesize a font if it is not already resident and cannot be located within the system font directory.

## Text Styles

Most text processing in Smalltalk-80 is performed not with instances of class StrikeFont in isolation but rather with instances of class TextStyle, whose properties include:

- fontArray (an Array of StrikeFonts)
- lineGrid (distance from top of text line to top of next text line)
- baseline (distance from top of text line to base of capital letters)
- additional lineGrids and baselines for lists and menus
- alignment code (flush left, flush right, centered, justified)
- indentation and tab stop parameters

These properties of a TextStyle, as its name implies, are mostly a matter of personal style and system convention. The fonts are usually members of a single family (although the system default, described below, violates this rule for historical reasons) in one or two sizes and several faces. The lineGrid (termed "leading" by typographers) is usually the height of the tallest font in the style plus a certain amount of additional white space. The baseline is shared by all of the fonts in the style so that the bases of their capital letters are aligned. Subscripts and superscripts, of course, would violate this rule, but they are not supported in this product release (although rudimentary capabilities do exist within classes StrikeFont, TextStyle, and DisplayScanner). Flush left alignment has historically been the default in Smalltalk-80, but other possibilities are certainly available.

Obvious uses of TextStyles include class ParagraphEditor and its subclasses (in Workspaces, System Transcripts, Projects, and the bottom panes of System Browsers, File Lists, and ChangeListViews). Less obvious uses include lists, menus, and title tabs of StandardSystemViews. Even less obvious uses include the String messages asParagraph and asDisplayText. This broad variety of uses prompts some common questions:

- What is the system default style?
- Can additional styles coexist?
- If so, how are they created and catalogued?

- How can the system default style be changed?
- How can the style of a view or subview be changed?

These questions are addressed in the following paragraphs.

The system default style is mentioned in chapter three of the Goldberg ("orange") book. It contains twenty-four fonts (two families, two sizes, and six faces) ordered as follows:

- Sans-Serif 10 (Basal, Bold, Italic)
- Serif 12 (Basal, Bold, Italic)
- Serif 10 (Basal, Bold, Italic)
- Sans-Serif 12 (Basal, Bold, Italic)
- all of the above repeated but Underlined

The original Smalltalk-80 system default style used the Xerox sans serif and serif font families. This product release maintains other characteristics of that style (including the unusual mixing of sans serif and serif families) but uses the Pellucida families in the default text style.

The new class TextStyleManager is a subclass of Dictionary and stores Associations between String names and TextStyles. A single instance of TextStyleManager is known as the global StyleManager. Particularly useful messages to this object include:

```
StyleManager inspect.
StyleManager
  styleName: aString
  fontNames: anArrayOfStrings
  lead: anInteger.
StyleManager
  styleName: aString
  baseNames: anArrayOfStrings
  lead: anInteger.
```

The inspect message opens a DictionaryInspector on StyleManager.

The styleName:fontNames:lead: and styleName:baseNames:lead: methods return a new style named aString. The fontNames: version accepts font names with arbitrary face codes in anArrayOfStrings whereas the baseNames: version accepts only Basal font names and imposes the following order on the fonts:

- (anArrayOfStrings at: 1) Basal
- (anArrayOfStrings at: 1) Bold
- (anArrayOfStrings at: 1) Italic
- (anArrayOfStrings at: 1) BoldItalic
- similar sequence(s) for other element(s) of anArrayOfStrings (if any)
- all of the above repeated but Underlined

The actual font array is obtained from FontManager via the fontNames: message thereby invoking the font loading and synthesizing mechanisms discussed above. The lead: parameter is the amount of additional white space to add to the height of the tallest font to obtain the lineGrid for the style. Both methods also install the new style in StyleManager for future reference.

Thus the expression that created the system default style is:

```
StyleManager
  styleName: 'Pellucida Default 10 and 12'
  fontNames: #(
    'PellucidaSans-Serif10'
    'PellucidaSans-Serif10B'
    'PellucidaSans-Serif10I'
    'PellucidaSerif12'
    'PellucidaSerif12B'
    'PellucidaSerif12I'
    'PellucidaSerif10'
    'PellucidaSerif10B'
    'PellucidaSerif10I'
    'PellucidaSans-Serif12'
    'PellucidaSans-Serif12B'
    'PellucidaSans-Serif12I'

    'PellucidaSerif10U'
    'PellucidaSerif10BU'
    'PellucidaSerif10IU'
    'PellucidaSerif12U'
    'PellucidaSerif12BU'
    'PellucidaSerif12IU'
    'PellucidaSans-Serif10U'
    'PellucidaSans-Serif10BU'
    'PellucidaSans-Serif10IU'
    'PellucidaSans-Serif12U'
    'PellucidaSans-Serif12BU'
    'PellucidaSans-Serif12IU')
  lead: 3.
```

A similar expression using the base name technique is found in the System Workspace:

```
StyleManager
  styleName: 'Pellucida Sans-Serif 12 and 14'
  baseNames: #('PellucidaSans-Serif12' 'PellucidaSans-Serif14')
  lead: 3.
```

These expressions illustrate two style conventions. The first suggests font family and size in the style name. Mixing sans serif and serif families in one style, preferably with the font ordering convention described in the Goldberg book, is connoted by the common font family name prefix (assuming there is one!) concatenated with the word "Default". Thus the original Smalltalk-80 style name would be "Xerox Default 10 and 12". Note that embedded spaces are encouraged in style names (unlike font names, which must be storable in the filing system). The second convention is the use of three additional pixels of leading in styles mixing two near sizes of fonts. Most text in the context of the style is expected to be in one of the smaller fonts.

Expressions similar to these can be found in files in the Smalltalk text style directory (*/smalltalk/textStyles*). These files store not styles but rather expressions that create styles; this distinction is suggested by the file suffix ".ws" (an abbreviation for ".workspace"). These files can be filed in if wholesale style acquisition is desired, or specific expressions can be executed to acquire specific styles. The System Workspace holds an expression that references this

directory to discard all existing fonts and styles, read in small fonts, and create a new default style (useful on systems with a small screen):

```
Compiler evaluate:
  ((Disk file: '/smalltalk/textStyles/pruneToPellucidaDefault08and10.ws')
    contentsOfEntireFile).
```

The StyleManager maintains the mapping from style names to styles for future reference when it is desired to change the system default style or the style of a view or subview. The expression:

```
StyleManager changeDefaultTextStyle.
```

pops up a menu of resident styles, waits for a style to be selected with any mouse button (or aborts if the button is released outside the menu), and then changes the default style to the selected style. This also rebuilds system menus and recomposes text in scheduled views and subviews in the current project. This capability can be added as the "style" entry of the middle button menu of class ScreenController by filing in:

*/smalltalk/fileIn/addTextStyleToSystemMenu.st*

A subset of this capability (propagate the selected style to the title tab and subviews of the current view) is available as the "style" entry of the right button menu of scheduled controllers. See *User Interface Changes.*

An even smaller subset of this capability (propagate the selected style only to the subview) can be added as the "style" entry of the middle button menu of class ParagraphEditor and several of its subclasses by filing in:

*/smalltalk/fileIn/addTextStyleToYellowButton.st*

The methods at:, at:put:, and removeKey: are useful for more primitive manipulation of StyleManager; the last two automatically update the menu of resident styles. Note that changing a style in StyleManager by itself usually has no effect on any text since styles are typically copied before use. An experimental style can be tested by adding it to StyleManager and then selecting it with the appropriate menu button.

The method initializeMenus rebuilds system menus. It references several lists that should be extended for applications with private menus.

## Miscellaneous

Class StrikeFont has new instance variables ascentForStdAsciiChars and descentForStdAsciiChars. These maintain the envelope of characters space (Ascii Decimal Equivalent 032) through tilde (ADE 126) for use by class TextStyle to compute styles for lists and menus (see below). A simple TextStyle can be constructed by sending asTextStyle to an instance of StrikeFont. Finally, the metaclass understands readAll: and readFrom:; the latter is used by class StrikeFontManager to load fonts from the filing system.

Class TextStyle has new instance variables lineGridForLists, baselineForLists, lineGridForMenus, and baselineForMenus. These support conversion of the style for lists and menus with the methods asListStyle and asMenuStyle. The method flushFonts has been removed.

Class PopUpMenu (and hence class ActionMenu) replaces instance variable font with textStyle. The private method labels:font:lines: has been replaced by labels:textStyle:lines:.

The metaclass understands labels:lines:alignment: so that non-centered alignments may be easily specified.

Within class ParagraphEditor, the control-x key formerly de-emphasized the current selection. Now control-x switches to a BoldItalic font (if possible), control-X switches to a non-BoldItalic font, and control-e de-emphasizes.

A new instance variable textStyle has been added to classes DisplayTextView, ListView, StandardSystemView, StringHolderView, SwitchView, and TextView. A method recomposeWithTextStyle: has been added to classes Paragraph and TextList. Class FileList has been modified not to cache menus in instance variables. Within pool dictionary TextConstants, DefaultLineGrid and DefaultBaseline have been removed; CtrlX and Ctrle have been added; and Ctrlx has been changed.

# IEEE Floating Point Numbers

The LOS interpreter includes primitives for many operations performed directly by the Motorola MC68881 Floating Point Coprocessor. The MC68881 conforms to IEEE-754 floating-point standards; however, Smalltalk conforms only to the representation specification of IEEE-754, not necessarily the implementation specification.

The new floating point primitives are capable of generating exceptional values which print as visible Smalltalk code. These values include: positive and negative infinity, denormalized numbers, and not-a-number. Because of these new values, a new protocol for testing has been added, new instance creation methods have been added, and some existing protocol has been changed.

New methods for testing an instance of Float include testing for infinity and testing for valid numerical representation. isPositiveInfinity and isNegativeInfinity return true for plus and minus infinity, respectively, while isInfinity returns true in either case. isNAN ("is Not A Number") returns true if an instance of Float does not contain a valid floating-point number representation. (Such a value is returned when dividing infinity by infinity, for instance.) isNormal returns true if an instance of Float is a valid, non-infinite floating-point value.

An instance of Float may now be instantiated to these exceptional values. Class methods negativeInfinity, positiveInfinity, and notANumber create instances of Float initialized with the appropriate exceptional value, while notANumber: creates an instance of Float in which the argument is stored in the mantissa and an all-ones bit pattern is stored in the exponent as the exceptional value.

The Float method printOn: now correctly interprets exceptional values, printing an evaluatable expression in each case. For example, executing print it in a workspace on 1.0e30 * 1.0e10 prints Float positiveInfinity or executing "print it" on 0.0 ln prints Float negativeInfinity.

Although the additional primitive methods represent an increase in functionality, and exceptional values are handled more completely, users of Float objects might want to protect themselves against the cascading effects of exceptional values. Whereas the SOS interpreter would generally pop up a notifier indicating a failed coercion, the LOS interpreter will continue evaluation using the exceptional value. The exceptional value will generally propagate through the expression, possibly making it difficult to locate the error.

Use of the new testing methods whenever there is a likelihood of generating an exceptional value is a good general coding practice.

# Lazy Mutation

In the SOS system, a change in definition of a class that caused the class's code to be recompiled also caused all instances of that class to be immediately converted to the new definition. The conversion process is called "mutation". Mutation involves a become: operation, which is a potentially expensive operation in the LOS implementation. Part of the optimization of virtual image code in the LOS system involved eliminating unnecessary become: operations. Since not all instances of a class will continue to exist (i.e., some will be garbage-collected), the LOS implementation only mutates instances of a class that are "used". It does this by "catching" messages to unmutated instances of this class and performing the mutation before actually sending the message. This mutation upon message send is known as lazy mutation. Subsequent messages to a mutated instance operate in a normal manner. Objects marked for mutation that never receive a message are not mutated and, therefore, do not use the become: operation.

In most cases, a user will see no difference between mutating all instances immediately and mutation upon message sends. However, a certain sequence of events might lead to undesirable and unexpected consequences:

```
"redefine ExampleClass by adding a new instance variable."
all ← ExampleClass allInstances.
all do: [:each | each initializeNewInstanceVariable].
```

In the SOS system, this sequence of events will redefine ExampleClass and mutate automatically all existing instances of the old ExampleClass to the new definition of ExampleClass. All the instances of the new ExampleClass are then collected and initialized.

In the LOS system, mutation of an existing instance to the new definition of ExampleClass will not occur until a message is sent to the instance. At no time does this example send a message to an instance of the old ExampleClass. Instead, this code collects all the instances of the new ExampleClass (there probably aren't any) and tries to initialize them. Here is an alternative technique that will work around this difficulty:

```
all ← ExampleClass allInstances.
"redefine ExampleClass by adding a new instance variable."
all do: [:each | each initializeNewInstanceVariable].
```

In this sequence of events, all instances of the old ExampleClass are collected and *then* the definition of ExampleClass is changed. Now all the instances can be initialized and in the process mutated to the new ExampleClass definition. This technique points out that the only straightforward way to collect all instances of a class is *before* it is redefined.

# Storing and Retrieving Objects on a File

The LOS image includes a mechanism for storing and retrieving object representations (including objects with circularities) on a file (or other character stream). This mechanism has two advantages over the original Smalltalk storeOn: mechanism. First, storeOn: does not work for objects that contain circularities; second, storeOn:'s output is meant to be read in by the compiler which limits the number of literals in an object to 64. Thus, storeOn: will not correctly handle all object structures.

The */smalltalk/conversion* directory contains an SOS version of this reading and writing mechanism called *structSOSPackage.st.*. Incorporate this package into your SOS image by filing

it in and using the methods below. This package also contains a copying mechanism discussed in the next section. This package may be used to transfer structures between LOS and SOS images.

## Using the Reading and Writing Mechanism

Four *visible* messages are defined to provide the writing or reading of objects to or from files or character-streams.

### To Write Structures:

someObject storeStructureOn: aStream.

Stores an object representation on a character stream aStream.

someObject storeStructureOnFile: aString.

Stores an object representation on a file named aString.

### To Read Structures:

Object readStructureFrom: aStream.

Answers an object defined by stream aStream.

Object readStructureFromFile: aString.

Answers an object defined on a file named aString.

These programs should allow object representations to be written or read to or from string format.

## Implementation Details

This mechanism maps objects based on ==–equality. If an object has a circular structure when written out, it will be circular when read back in. Similarly, acyclic structures are read back in as acyclic structures. There are a few cautions, however:

1. There are some objects, such as processes, that may cause unexpected behavior if an attempt is made to write them out, or particularly to read back in. Contexts are treated specially in that the sender is always written out as nil. CompiledMethods are written out in a special format, which is compatible with both SOS and LOS images. Also be aware that the receiver of a MethodContext in which the block context was created is also copied as part of the definition of the MethodContext.
2. Smalltalk treats certain objects in a special way, guaranteeing their uniqueness. A new selector, isUniqueValue, has been defined that returns a boolean value, stating whether the object has this property. Such classes include UndefinedObject, Boolean, Symbol, SmallInteger and Character. Objects in these classes are mapped to the corresponding

object in the target image. Floating point numbers are written out to 9 digits of accuracy. If more (or less) accuracy is desired, it is necessary to modify the method Float printStructureOn:.

3. *This step does not apply to objects for which* isUniqueValue *is true.* Objects that correspond to global Smalltalk names in the original image are mapped to objects with corresponding global Smalltalk names in the target image. This prevents classes and metaclasses from being duplicated. It requires, however, that you be responsible for ensuring that the target image defines all global variables that are referenced (directly or indirectly) by the object in the source image. If two Smalltalk globals refer to the same object, the result is nondeterministic.
4. Most classes are stored and read in using methods inherited from class Object, which copy instance variables and array elements using instVarAt:, etc. This means that classes must have identical definitions in both the original and target images. It also means that classes that depend on the hash values should be handled specially. Currently, only Set and its subclasses are treated specially for this reason. String and Number (and their subclasses) are treated specially for conciseness of notation (and because SmallInteger must be treated specially anyway). CompiledMethod is also treated specially to ensure the transfer between SOS and LOS images.
5. A receiver's dependents (from the Smalltalk dependency mechanism) are not mapped.

# Copying Circular Structures

The following methods implement a mechanism for copying Smalltalk objects that may contain circularities. The Smalltalk method shallowCopy does not generally copy the complete structure, while deepCopy generally only works for non-circular structures.

## Using the Copying Mechanism

Two *visible* messages are defined to provide the copying of structures.

someObject structureCopy

Answers a copy of the object.

someObject structureCopyWithDict: anIdentityDictionary

Answers a copy of the object, given that a partial list of mappings from objects in the old domain to the new are in anIdentityDictionary. The method may have side effects on anIdentityDictionary, adding new mappings.

The simplest way to use these methods is to use structureCopy. However, if you want to have a handle on the mapping dictionary (either to pre-specify some mappings, to know the mappings after the copy has been created, or to get a copy of several objects that may have common subobjects), you should supply your own IdentityDictionary and use structureCopyWithDict:.

## Implementation Details

This mechanism maps objects based on ==\quality. There are a few cautions, however:

1. The copying of objects such as processes will probably cause strange behavior. When a context is copied, the sender field in the new context is nil. The receiver part of a MethodContext, however, becomes mapped to a new object just as any other object would. CompiledMethods are not copied; rather, the original object is returned. The idea here is that compiled methods should be *constant* objects.
2. Smalltalk treats certain objects in a special way, guaranteeing their uniqueness. These objects in classes such as Boolean, SmallInteger, and Character will return themselves rather than a copy.
3. Most classes are stored and read in using methods inherited from class Object, which copy instance variables and array elements using instVarAt:, etc. This means that classes that depend on the hash values should be handled specially. Currently, only Set and its subclasses are treated specially for this reason.
4. A receiver's dependents (from the Smalltalk dependency mechanism) are not mapped.

# LOS System Workspace Modifications

The LOS System Workspace has additional text that describes some of the added functionality of the LOS Smalltalk system. The list of globals now includes FontManager, an instance of StrikeFontManager, which maps names to instances of StrikeFont, and, StyleManager, an instance of TextStyleManager, which maps names to instances of TextStyle. The list also includes OSEnvironmentVariables, a Dictionary containing the environment variables passed to the Smalltalk interpreter.

A new section in the System Workspace is titled *Fonts and Text Styles*. This section includes:

```
StyleManager inspect
```

Opens an inspector on all the text styles in the image.

```
StyleManager
    styleName: 'Pellucida Sans-Serif 12/14'
    baseNames: #('PellucidaSans-Serif12' 'PellucidaSans-Serif14')
    lead: 3.
```

Installs a new text style containing two fonts. This text style is named 'Pellucida Sans-Serif 12/14'. If the fonts are not contained in the image, they will be loaded from the */fonts* directory. The vertical spacing (leading) for this text style is 3 pixels. The text style contains basal, bold, italic, and bold italic faces for each font.

```
StyleManager changeDefaultTextStyle
```

Chooses a new default text style from a menu.

```
Compiler evaluate: ((Disk file: '/smalltalk/textStyles/
pruneToPellucidaDefault08and10.ws') contentsOfEntireFile).
```

Creates a new TextStyle in the Xerox style with mixed serif and sans serif fonts. Discard all other TextStyles and StrikeFonts.

These are additions to the *Display* section:

```
DisplayScreen displayExtent: 1376@1024
```

Sets the Smalltalk DisplayScreen size to that of the 4406 visible display screen size.

```
Display setMouseBounds: (-50@-50 corner: 1500@1500)
```

Allows the mouse cursor to move outside the visible screen bounds.

The *Measurements* section has a removed comment, "takes a long time", with respect to core, oopsLeft, and coreLeft because these are now fast measurements. The section also has two new expressions:

```
Smalltalk garbageCollect
```

A garbage collection through all object space is initiated by evaluating this expression.

```
TekSystemCall execSystemUtility: '/bin/free' withArgs: (OrderedCollection with:
'/dev/disk'
```

Asks the operating system how much space is available on the hard disk.

## Miscellaneous Changes

Here is a list of some visible changes to the image not mentioned in any other section:

- The global variable SourceFiles now can have 8 elements instead of being limited to 4.
- The global variable Environment has been renamed OSEnvironmentVariables.
- The use of the writeCloneWithout: message to a system tracer produces a clone but does not produce new source files. New source files may be produced in a separate step.

# SMALLTALK DIRECTORIES

With this release of Smalltalk, some new directories and additions and changes to existing directory files have been made.

## New Directories

The directory */smalltalk/conversion* has been added. This contains files for converting one version of a Smalltalk image file into another. These mostly relate to converting SOS (Small Object Space) to LOS (Large Object Space) image files.

A special directory, */smalltalk/demo/forms*, has been created for forms alone.

New text styles have been added to this release of Smalltalk. Thus, the directory, */smalltalk/textStyles*, has been created and contains code to create instances of text styles in an image. Specifically,

| | |
|---|---|
| PellucidaDefault08and10.ws | Contains code to install the small default style in the Xerox manner, that is, basal, bold, and italic (a triplet) in addition to mixed serif and sans serif faces. |
| PellucidaDefault10and12.ws | Contains the medium sized default faces. |
| PellucidaSans-Serif08tight.ws | Contains an example of minimal vertical spacing. |
| PellucidaSans-Serif.ws | This file along with PellucidaSerifs.ws contain code to create all available text styles in the quadruplet format, that is, basal, bold, italic, and bold italic. |
| PellucidaTypewriters.ws | Contains code to create monospaced fonts. |
| example.wsf | Contains code to create a single, large text style in the quadruplet format. |
| pruneToPellucidaDefault08and10.ws | Contains code to remove all text styles and create the small default triplet style. |

## New Files

The following files in the directory, /smalltalk/fileIn, have been added:

- Graphics-Fractals.st
- Mastermind-Support.st
- addTextStyleToSystemMenu.st
- addTextStyleToYellowButton.st
- extendedBrowser.st
- joydiskAccessAndExample.st
- timedMethods.st
- workspaceFileOut.st

# PRINTING SMALLTALK BITMAP FILES

Look in the */samples/printer* directory for a C program, *bprint.c*, that prints Smalltalk forms or bitmaps on a Tektronix 4644 printer. You can either use this program as it stands if you have the 4644 printer or you can modify the program to be compatible with a different printer.

This program, *bprint.c*, prints Smalltalk bitmaps as generated by the screenCopy menu item or from a fileOut of a specific form. If you modify the program, the default graphic density and

screen width pixels per printer line should be determined by the characteristics of your printer. In *bprint.c*, the default graphic density is double. Option "+s" enables single-density mode which gives you a larger image but with possible truncation.

# Appendix A
# Conversion of SOS to LOS Images

This appendix shows you how to transfer work that you have done in a Small Object Space (SOS) image (on the 4404 system) to the Large Object Space (LOS) system on a 4405 or 4406 system.

Once you have an image, myImage, which you have created on a 4404 machine, transferred to a 4406 machine, you simply type the familiar:

*smalltalk myImage*

To make it convenient for you, Smalltalk has been modified to recognize whether your image is a SOS or a LOS image. (Of course, if you just type smalltalk by itself on the command line on a 4406 machine, you will invoke the default LOS interpreter.)

## How to Convert an Image

There are three ways to convert a SOS to a LOS image:

1. Technique 1: Use fileIn and fileOut to file code out of the SOS image and into the LOS image.
2. Technique 2: Use LOSConversionTracer to convert a SOS image into a LOS image.
3. Technique 3: Use the Changes file you have maintained to file in changes to a LOS image.

Technique 1 is preferred over the other two techniques because it is most likely to result in a cleanly functioning LOS image, and it is conceptually the simplest. However, if you have made a lot of modifications to your image, Technique 1 may be time-consuming, since classes and methods that you have added or modified have to be filed out of the SOS image and filed in to the LOS image.

Technique 2 is preferred over Technique 3 mainly because it is simpler. However, to use Technique 2 successfully, your image should satisfy both of the following conditions:

- Your SOS image must be "clean". This means that your image must be free of uncollectible garbage, such as undeclared objects, hanging DoIts, obsolete classes, obsolete associations, etc. These mainly involve pointers that point to no longer existing objects. You may have created uncollectible garbage by doing operations like control-C'ing while doing a fileIn operation, removing a class from the system while still having a class variable assigned to it, and so forth. See later under Technique 2 *Eliminating Uncollectible Garbage*.
- Your image must be small enough to accommodate and execute the LOSConversionTracer.

Technique 3 is the least preferred because it is tedious, time-consuming, and error prone. You should only choose this technique if the other two do not work.

# The Conversion Procedures

Choose one of the following three techniqes, read through the whole discussion of the procedure, then perform the procedure.

## Technique 1

1. Bring up the SOS image that you want to convert.
2. Transfer code by filing out all changes that you want to transfer to an LOS image. You can use the following template from the System Workspace:

   (FileStream newFileNamed 'fileName.st') fileOutChanges

   Make sure that you execute this expression in each project including the top project.

   *NOTE*

   *If you have filed out a class from the browser, it is removed from the* ChangeSet *and, thus, it must be filed out separately.*

3. If you have *objects* you want to transfer to an LOS image, fileIn */smalltalk/conversion/structSOSPackage.st* and write out the structures. See *Storing and Retrieving Objects on a File* in these *Notes*.
4. Bring up the standard, default LOS image.
5. Create a personal Changes file. For how to do this, see *Installing Your Own Image* in the *Introduction to Smalltalk* manual.
6. Use a ChangeListView, eliminate unnecessary items, and incorporate the remaining code into your image by using the middle button "doit" item. Alternatively, fileIn all the files created by the fileOut procedure in the SOS image. (If you would like to check on the filed in code, you can do an Undeclared inspect after the filing in operation. If you find any references, look at the relevant code in the LOS image and fix the code.)
7. Read in the structures you may have written out in step 3. See *Storing and Retrieving Objects on a File* in these *Notes*.
8. Snapshot the new LOS image to create a personal image.

## Technique 2

There are two parts to this procedure:

1. Eliminate uncollectible garbage from your SOS image.
2. File in the conversion methods, which include LOSConversionTracer, execute the tracer, and file in the rest of the conversion methods.

## Eliminating Uncollectible Garbage

Although a procedure is presented here to describe how to clean up your image, you should be aware that cleaning up your image is more an art than a science. The following procedure will help you track down uncollectible garbage; however, it is not guaranteed to find all uncollectible garbage. Futhermore, once these expressions reveal the existence of some garbage, you must manually trackdown and eliminate the garbage yourself.

Many of the following Smalltalk expressions are found in the system workspace. You can execute these expression in either the system workspace or in an ordinary workspace. Be sure the system transcript is open since some of the expressions write text to the system transcript and you may want to save this text. In the following procedure, to **do** something means to select it and perform a DoIt menu operation, while to **print** something means to select it and perform a PrintIt menu operation.

1. **Do** Smalltalk forgetDoIts. This eliminates hanging "DoIts", which, for example, you may have created by doing a control-C while executing code.
2. **Do** Checker allUnscheduledDependentViews do: [:aView | aView release]. You may have some unscheduled windows that are still referenced via the dependency mechanism. This releases unscheduled windows.
3. **Print** (Object classPool at: #DependentFields) keys. This prints a set of object dependents, some of which may be garbage. There should typically be an object for each open view, including those in other projects. Typical ones are TextCollector (system transcript), an InfiniteForm (the background), various workspaces, various browsers, etc. If you have garbage dependents, execute the next code in the system workspace with the argument to isKindOf: the appropriate class to release only the garbage desired. This expression may have to be executed more than once.
4. **Do** Undeclared inspect. The resultant inspector should normally be empty. If not, check for references and remove or declare as appropriate.
5. **Print** Checker obsoleteClasses. This should result in an empty OrderedCollection. If not, use Smalltalk collectPointersTo: to eliminate the references to each obsolete class. Make sure you save the result. You may also want to use Checker obsoleteAssociations in the same manner to help eliminate obsolete classes.
6. **Do** Checker rehashBadSets. This verifies that keys for sets are valid. The system transcript prints a message saying how many sets had to be rehashed.
7. **Print** the following code:

```
Smalltalk classNames select:
  [:x| (Smalltalk at: x) superclass
  class ~~(Smalltalk at: x) class superclass]
```

8. **Print** the following code:

```
"Check for missing classes in subclass lists."
Smalltalk allBehaviorsDo: [:class | sClass ← class superclass.
  sClass notNil ifTrue: [(sClass subclasses includes: class)
    ifFalse: [Transcript show: class printString,
    ' is missing from superclass ', sClass printString]]].
```

9. **Print** the following code:

```
"Check for duplicate or erroneous classes in subclass lists."
Smalltalk allBehaviorsDo: [:class | subs ← class subclasses.
  subs do: [:each |
    (each superclass ~~ class)
      ifTrue: [Transcript show: each printString,
      ' is incorrectly duplicated in subclass list of ',
      class printString; cr.  class removeSubclass: each]]].
```

10. **Print** the following code:

```
"Check for classes in subclass lists which are not contained
 in the system dictionary."
Smalltalk  allBehaviorsDo: [:class |  class subclasses do: [:each |
  ((Smalltalk at: each name ifAbsent: [nil]) = nil and: [each isMeta not])
    ifTrue: [class removeSubclass: each.
      Transcript show: 'Removing subclass ', each printString.
      ' from ', class printString; cr]]]
```

Classes that have incomplete or incorrect references within the class hierarchy either are not written in the clone image or cause the tracer process to break. Before using the LOSConversionTracer, fix the class hierarchy in the image based on information from steps 8, 9, and 10.

## Filing In and Using the LOSConversionTracer

Before you go through the following procedure, be sure that you have a clean image; that is, make sure that you have gone through the procedure described under the heading *Eliminating Uncollectible Garbage* earlier.

1. Bring up your SOS image using the *+m=3000* option.

   smalltalk +m=3000 <yourSOSImage>

2. If you have a T2.1.2 version SOS image, file in the file *deltaT2.1.2ToT2.1.2a.st* from the */smalltalk/conversion* directory. This brings your image up to the level expected by the rest of this conversion technique.

3. Now file in the file:

   */smalltalk/conversion/preCloneT2.1.2aToT2.2.0.st*

   (Note that this file must *only* be used with a SOS image!)

4. Make a snapshot and continue.
5. Make the new image by opening a workspace and performing a DoIt menu operation on:

   ```
   LOSConversionTracer writeCloneWithout: (Set with: SystemTracer
   with: TekSystemTracer with: LOSConversionTracer).
   ```

   The conversion method asks you to name the new image.
6. When the cloning is complete, exit the SOS interpreter by quitting without saving the image.
7. Bring up the new image by typing the name you gave it in step 4.
8. Important Note: The following file must be filed in *just once and only once* in a newly created image converted with the *preCloneT2.1.2aToT2.2.0.st* file! Now file in the file:

   */smalltalk/conversion/postCloneT2.1.2aToT2.2.0.st*
9. Make a snapshot. The image produced should be functionally equivalent to the original SOS image, but now is compatible with the LOS interpreter. You may now treat this image as if it were an ordinary LOS image.

## Technique 3

1. Bring up the standard, default LOS image.
2. Create a personal Changes file. For how to do this, see *Installing Your Own Image* in the *Introduction to Smalltalk* manual.
3. Read the old SOS Changes file into a ChangeListView using the middle button "fileIn" menu item.
4. Select items and perform a "DoIt" from the middle button menu to incorporate appropriate method definitions, class definitions, etc., into your image.
5. Make a snapshot.

Note that the sequence of additions ("DoIts") may affect the result. It is recommended that you incorporate the class definitions first, check for and remove duplicates of items, and then incorporate method definitions.

# Appendix B
# Smalltalk-80 Version T2.2.0 Files

The following is a list of all the files associated with Smalltalk-80 Version T2.2.0 system.

Directory */smalltalk*

standardImage

Directory */smalltalk/conversion*:

deltaT2.1.2ToT2.1.2a.st
postCloneT2.1.2aToT2.2.0.st
preCloneT2.1.2aToT2.2.0.st
structSOSPackage.st

Directory */smalltalk/demo*:

Othello.script
Othello.st
Pentominos.script
Pentominos.st
README
Robots.script
Robots.st
WaterJugs.st
demoChanges
demoImage
makingADemoImage.ws
steps18-25.robot
steps27-33.robot

Directory */smalltalk/demo/forms*:

aim.form
fall1.form
fall2.form
fall3.form
head1.form
head2.form
head3.form
head4.form
head5.form
laundry1.form
laundry2.form
laundry3.form
laundry4.form
laundry5.form
man1.form
man2.form
man3.form
man4.form
man5.form
man6.form

man7.form
man8.form
man9.form
pegasus.form
pendulum1.form
pendulum10.form
pendulum11.form
pendulum12.form
pendulum13.form
pendulum2.form
pendulum3.form
pendulum4.form
pendulum5.form
pendulum6.form
pendulum7.form
pendulum8.form
pendulum9.form
sketch.form
tekLogo.form
usa.form
waterfall.form

Directory */smalltalk/fileIn*:

Animation.st
BookIndexBrowser.st
Clock.st
Examples-Subtasking.st
FinancialHistory.st
Formclass-readMacPaintFile:.st
Graphics-Fractals.st
IconPopUpMenu.st
KineticGraphics.st
Mastermind-Support.st
PointingHand.st
PopUpMenuHelp.st
ProjectBrowser.st
ProtocolBrowser.st
README
Signals-Support.st
Sound-Support.st
WireList-ASimpleMVCExample.st
addTextStyleToSystemMenu.st
addTextStyleToYellowButton.st
backgroundForm.st
blueInspect.st
corePlot.ws
extendedBrowser.st
findClass.st
hardCopyFunctionKey.st
inspectIt.st

joydiskAccessAndExample.st
sampleBook.index
slideMaker.st
symbolRecovery.st
timedMethods.st
toothpaste.ws
workspaceFileOut.st
zoomTo.st

Directory */smalltalk/system*:

smalltalk16
standardChanges.VersionT2.2.0
standardImage16
standardSources.VersionT2.2.0

Directory */smalltalk/system/initialization*:

SOSSystemWorkspace.ws
SystemWorkspace.ws
black.form
block.form
borderform.form
curve.form
darkgray.form
erase.form
gray.form
in.form
installPellucidaDefault10and12TextStyle.st
lightgray.form
line.form
magnify.form
out.form
over.form
repeatcopy.form
reverse.form
select.form
singlecopy.form
specialborderform.form
togglegrids.form
under.form
white.form
xgrid.form
ygrid.form

Directory */smalltalk/textStyles*:

PellucidaDefault08and10.ws
PellucidaDefault10and12.ws
PellucidaSans-Serif08tight.ws
PellucidaSans-Serifs.ws
PellucidaSerifs.ws
PellucidaTypewriters.ws

XeroxDefault10and12.ws
example.ws
pruneToPellucidaDefault08and10.ws

# Appendix C
# Changes in the Smalltalk-80 Images

The Tektronix 4406 Smalltalk system supports two complete Smalltalk images, the Large Object Space image (version T2.2.0) and the Small Object Space image (version T2.1.3). The following lists detail the differences between these images and the Smalltalk image previously released on the Tektronix 4404 (version T2.1.2a).

In the left column, you will find the classes or methods that are changed between that image and the 4404 version T2.1.2a. The right column shows how this image was changed. An asterisk (*) in the right column shows that this change affects both the Large Object Space and the Small Object Space image. If no asterisk is present, the change concerns only the image being discussed.

## LARGE OBJECT SPACE DIFFERENCES

The following list details the difference between the Tektronix Large Object Space image, version T2.2.0, and the Small Object Space image, version T2.1.2a.

| | |
|---|---|
| define Behavior | modified |
| Behavior class initializeObsoleteDictionaries | new |
| Behavior class obsoleteClassDictionary | new |
| Behavior class obsoleteMetaclassDictionary | new |
| | |
| Behavior allInstances | modified |
| Behavior allInstancesDo: | modified |
| Behavior compileUnchecked: | modified |
| Behavior obsoleteForMutationTo: | new |
| Behavior recompile:from: | modified |
| Behavior removeSelectorSimply: | modified* |
| Behavior subclassesForMutation | new |
| Behavior whichSelectorsReferTo:special:byte: | modified |
| | |
| Benchmark testCompiler | modified |
| BitEditor class initialize | modified* |
| BlockContext adjustPCsForStructReading | new |
| BlockContext adjustPCsForStructWriting | new |
| | |
| Boolean isUniqueValue | new |
| Boolean structureCopyWithDict: | new |
| Browser removeClass | modified* |
| Browser renameClass | modified* |
| | |
| define BytecodeArray | new |
| BytecodeArray become: | new |
| BytecodeArray initialPC | new |

| | |
|---|---|
| ChangeScanner scanClassExpression:do: | modified* |
| Character class readDefinitionFrom:map: | new* |
| Character isUniqueValue | new |
| Character storeDefinitionOn:auxTable: | new |
| Character structureCopyWithDict: | new |
| | |
| Checker class classVariablesNotReferenced | new* |
| Checker class findCorruptedSourceCode | modified |
| Checker class printClassVariablesNotReferencedOn: | new* |
| | |
| Class nonVariableSubclass:instanceVariableNames:c... | new* |
| Class obsolete | modified |
| Class obsoleteForMutationTo: | new |
| Class replaceNameWith: | new* |
| Class validateFrom:in:instanceVariableNames:methods: | modified |
| | |
| ClassDescription compile:classified:notifying: | modified |
| ClassDescription definition | modified* |
| ClassDescription kindOfSubclass | modified* |
| ClassDescription moveChangesToSources: | new* |
| ClassDescription obsoleteForMutationTo: | new |
| ClassDescription updateInstancesFrom: | modified |
| ClassDescription validateFrom:in:instanceVariableN... | modified |
| | |
| Collection class initialize | modified |
| Collection growSize | modified* |
| Collection maxSize | modified* |
| | |
| define CompiledMethod | new |
| | |
| CompiledMethod class initialize | new |
| CompiledMethod class newBytes:flags:nTemps:nArgs:nS... | new |
| CompiledMethod class nullSourceDescriptor | new |
| CompiledMethod class quickReturnPC | new |
| CompiledMethod class toReturnField: | new |
| CompiledMethod class toReturnSelf | new |
| CompiledMethod class sourceDescriptorForFile:P... | new |
| | |
| CompiledMethod action | new |
| CompiledMethod at: | new |
| CompiledMethod at:put: | new |
| CompiledMethod cacheTempNames: | new |
| CompiledMethod endPC | new |
| | |
| CompiledMethod fieldsTouched | new |
| CompiledMethod fileIndex | new |
| CompiledMethod flags | new |

| | |
|---|---|
| CompiledMethod frameSize | new |
| CompiledMethod getSource | new |
| | |
| CompiledMethod header | new |
| CompiledMethod header:instructions: | new |
| CompiledMethod initialPC | new |
| CompiledMethod instructions | new |
| CompiledMethod isQuick | new |
| | |
| CompiledMethod isReturnField | new |
| CompiledMethod isReturnSelf | new |
| CompiledMethod last | new |
| CompiledMethod literalAt: | new |
| CompiledMethod literalAt:put: | new |
| | |
| CompiledMethod literals | new |
| CompiledMethod literals: | new |
| CompiledMethod messages | new |
| CompiledMethod needsLargeFrame | new |
| CompiledMethod needsStack:encoder: | new |
| | |
| CompiledMethod nullSourceDescriptor | new |
| CompiledMethod numArgs | new |
| CompiledMethod numLiterals | new |
| CompiledMethod numStack | new |
| CompiledMethod numTemps | new |
| | |
| CompiledMethod numTempsField | new |
| CompiledMethod objectAt: | new |
| CompiledMethod objectAt:put: | new |
| CompiledMethod openByteCodeStream | new |
| CompiledMethod primitive | new |
| | |
| CompiledMethod putSource:class:category:inFile: | new |
| CompiledMethod putSource:inFile: | new |
| CompiledMethod readsField: | new |
| CompiledMethod readsRef: | new |
| CompiledMethod refersToLiteral: | new |
| | |
| CompiledMethod returnField | new |
| CompiledMethod scanFor: | new |
| CompiledMethod scanLongLoad: | new |
| CompiledMethod scanLongStore: | new |
| CompiledMethod setPrimitive: | new |
| | |
| CompiledMethod setSourcePosition:inFile: | new |
| CompiledMethod setTempNamesIfCached: | new |

| | |
|---|---|
| CompiledMethod size | new |
| CompiledMethod sourceCode | new |
| CompiledMethod sourceDescriptor | new |
| CompiledMethod sourceDescriptor: | new |
| CompiledMethod sourceOffset | new |
| CompiledMethod symbolic | new |
| CompiledMethod trailerSize | new |
| CompiledMethod useLargeFrame | new |
| CompiledMethod who | new |
| CompiledMethod writesField: | new |
| CompiledMethod writesRef: | new |
| Compiler evaluate:in:to:notifying:ifFail: | modified* |
| ContextPart class basicNew: | new* |
| ContextPart class new: | new* |
| ContextPart class readDefinitionFrom:Map: | new* |
| ContextPart at:put: | modified* |
| ContextPart copy | new* |
| ContextPart doPrimitive:receiver:args: | modified |
| ContextPart instVarAt:put: | modified |
| ContextPart stackp: | modified |
| ContextPart storeDefinitionOn:auxTable: | new |
| ContextPart structureCopyWithDict: | new |
| ContextPart tryPrimitiveFor:receiver:args: | modified |
| ControlManager discardCachedDisplayForms | new* |
| ControlManager restore | modified* |
| Cursor class currentCursor: | new* |
| Cursor centerCursorInViewport | modified* |
| Debugger class context: | modified |
| Debugger bindingOf:forStore: | modified* |
| Debugger pcRange | modified |
| Debugger spawnEdits:from: | modified |
| Debugger step | modified |
| Dictionary storeDefinitionOn:auxTable: | new |
| DisplayBitmap class basicNew: | new* |
| DisplayBitmap class maxSize | new |
| DisplayBitmap class new: | new* |

| | |
|---|---|
| DisplayScreen class currentDisplay: | modified* |
| DisplayScreen class displayExtent: | modified* |
| DisplayScreen resetFrom:extent: | modified* |
| DisplayScreen resetFrom:extent:offset: | modified* |
| DisplayScreen setDisplayStateFrom: | new* |
| DisplayScreen setMouseBounds: | modified* |
| DisplayScreen setMouseBoundsUpper:low... | modified* |
| DisplayScreen viewport | new* |
| DisplayScreen viewportCenter | new* |
| | |
| DisplayText class text: | modified* |
| DisplayText textStyle | new* |
| | |
| define DisplayTextView | modified* |
| DisplayTextView initialize | modified* |
| DisplayTextView textStyle | new* |
| DisplayTextView textStyle: | new* |
| | |
| Encoder noteSourceRange:forNode: | modified* |
| Encoder sourceMap | new* |
| Encoder sourceMap: | modified* |
| Encoder tempNames | modified* |
| | |
| False class readDefinitionFrom:map: | new |
| FileDirectory fullName | modified* |
| | |
| FileList createDirectory | modified* |
| FileList createFile | modified* |
| FileList directoryMenu | modified* |
| FileList fileListMenu | modified* |
| FileList fileName: | new* |
| FileList newFileMenu | modified* |
| FileList resetFileMenu | modified* |
| | |
| FileStream class initialize | modified* |
| FileStream appendFileStream: | new* |
| FileStream binary | modified* |
| FileStream contentsOfEntireFile | modified* |
| FileStream nextPutAll: | modified* |
| FileStream nextPutAll:startingAt: | new* |
| FileStream nextPutAll:startingAt:to: | new* |
| FileStream padTo: | modified* |
| FileStream size | modified* |
| FileStream text | modified* |
| | |
| FillInTheBlank class example1 | modified* |
| FillInTheBlank class example2 | modified* |

| | |
|---|---|
| define Float | modified* |
| Float class initialize | modified* |
| Float class negativeInfinity | new* |
| Float class notANumber | new* |
| Float class notANumber: | new* |
| Float class positiveInfinity | new* |
| | |
| Float arcCos | modified* |
| Float arcSin | modified* |
| Float arcTan | modified* |
| Float exp | modified* |
| Float floorLog: | modified* |
| | |
| Float isInfinity | new* |
| Float isNAN | new* |
| Float isNegativeInfinity | new* |
| Float isNormal | new* |
| Float isPositiveInfinity | new* |
| | |
| Float ln | modified* |
| Float log | modified* |
| Float printOn: | modified* |
| Float printStructureOn: | new |
| Float truncated | modified* |
| | |
| Form class readFormFile: | modified* |
| FormHolderView cancel | modified* |
| | |
| InputSensor currentCursor: | modified* |
| Inspector acceptText:from: | modified |
| Inspector fieldList | modified* |
| | |
| define ListView | modified* |
| ListView initialize | modified* |
| ListView list: | modified* |
| ListView selectionBox | modified* |
| ListView textStyle | new* |
| ListView textStyle: | new* |
| | |
| define LiteralArray | new |
| | |
| LiteralArray class new:instructions: | new |
| LiteralArray become: | new |
| | |
| MessageNode emitForEffect:on: | modified* |

| | |
|---|---|
| MessageNode emitForValue:on: | modified* |
| Metaclass obsoleteForMutationTo: | new |
| Metaclass resetSuperclassToMetaclassForMutation | new |
| Metaclass resetSuperclassToNilAfterMutation | new |
| Metaclass storeDefinitionOn:auxTable: | new |
| | |
| MethodContext adjustPCsForStructReading | new |
| MethodContext adjustPCsForStructWriting | new |
| MethodContext at:put: | modified* |
| MethodContext basicAt:put: | modified* |
| MethodContext setSender:receiver:method:arguments: | modified* |
| | |
| MethodDefinitionChange accept:notifying: | modified* |
| MethodDefinitionChange sourceFileAndPosition: | modified* |
| | |
| define MethodDictionary | new |
| | |
| MethodDictionary class new | new |
| MethodDictionary class new: | new |
| | |
| MethodDictionary basicAt: | new |
| MethodDictionary basicAt:put: | new |
| MethodDictionary basicSize | new |
| MethodDictionary become: | new |
| | |
| MethodDictionary copy | new |
| MethodDictionary grow | new |
| MethodDictionary growSize | new |
| MethodDictionary keyArray | new |
| MethodDictionary rehash | new |
| | |
| MethodDictionary removeDangerouslyKey:ifAbsent: | new |
| MethodDictionary removeKey:ifAbsent: | new |
| MethodDictionary setTally: | new |
| MethodDictionary shallowCopy | new |
| MethodDictionary valueArray | new |
| | |
| MethodNode generate: | modified* |
| MethodNode generateNoQuick | modified* |
| MethodNode sourceMap | modified* |
| | |
| NotifierView class openContext:label:contents: | modified* |
| NotifierView class openInterrupt:onProcess: | modified* |
| NotifierView textStyle: | new* |
| Number class readFrom: | modified* |
| Number printStructureOn: | new |

| | |
|---|---|
| Number storeStructureOn:auxTable: | new |
| Object class readDefinitionFrom:map: | new |
| Object class readFixedDefinitionFrom:map:value: | new |
| Object class readStructureFrom: | new |
| Object class readStructureFrom:map: | new |
| Object class readStructureFromFile: | new |
| Object isUniqueValue | new |
| Object shallowCopy | modified* |
| Object storeDefinitionOn:auxTable: | new |
| Object storeStructureOn: | new |
| Object storeStructureOn:auxTable: | new |
| Object storeStructureOnFile: | new |
| Object structureCopy | new |
| Object structureCopyWithDict: | new |
| Paragraph recomposeWithTextStyle: | new* |
| ParagraphEditor class initialize | modified* |
| ParagraphEditor changeEmphasis: | modified* |
| ParagraphEditor emphasisDefault:keyedTo: | modified* |
| ParagraphEditor readKeyboard | modified* |
| Pen mandala:diameter: | modified* |
| PipeReadStream contentsOfEntireFile | modified* |
| Point negated | new* |
| define PopUpMenu | modified* |
| PopUpMenu class labels:lines: | modified* |
| PopUpMenu class labels:lines:alignment: | modified* |
| PopUpMenu labels:textStyle:lines: | new* |
| PopUpMenu markerTop: | modified* |
| PopUpMenu rescan | modified* |
| PopUpMenu reset | modified* |
| PositionableStream through: | modified* |
| PositionableStream upTo: | modified* |
| define ProcessorScheduler | modified* |
| ProcessorScheduler class initialize | modified* |
| ProcessorScheduler absolutelyTheHighestPriority | new* |
| ProcessorScheduler executeWithoutPreemption: | new* |
| ProcessorScheduler highestPriority: | modified* |
| ProcessorScheduler resetPriorities | modified* |
| Project enter | modified* |

| | |
|---|---|
| ProjectController class initialize | new* |
| Rectangle class fromUser: | modified* |
| Rectangle negated | new* |
| ReturnNode emitForReturn:on: | modified* |
| ReturnNode emitForValue:on: | modified* |
| ReturnNode pc | modified* |
| Scanner scanFieldNames: | modified* |
| ScreenController forkOSshell | modified* |
| ScrollController canScroll | modified* |
| ScrollController canScrollDown | new* |
| ScrollController canScrollUp | new* |
| ScrollController controlInitialize | modified* |
| ScrollController moveMarker | modified* |
| ScrollController moveMarker: | modified* |
| ScrollController scrollDown | modified* |
| ScrollController scrollUp | modified* |
| SequenceableCollection hash | modified* |
| Set class maxSize | new* |
| Set class new | modified* |
| Set class readDefinitionFrom:map: | new |
| Set storeDefinitionOn:auxTable: | new |
| Set structureCopyWithDict: | new |
| SmallInteger structureCopyWithDict: | new |
| SortedCollection class readDefinitionFrom:map: | new |
| SortedCollection structureCopyWithDict: | new |
| StandardSystemController class initialize | modified |
| StandardSystemController textStyle | new* |
| StandardSystemController textStyle: | new* |
| StandardSystemView | redefined old class* |
| StandardSystemView getFrame | modified* |
| StandardSystemView initialize | modified* |
| StandardSystemView label: | modified* |
| StandardSystemView label:style: | new* |
| StandardSystemView resetLabel: | modified* |
| StandardSystemView resetLabel:style: | new* |
| StandardSystemView textStyle | new* |
| StandardSystemView textStyle: | new* |
| Stream do: | modified* |
| Stream nextPutAll:startingAt:to: | new* |

| | |
|---|---|
| define StrikeFont | modified* |
| StrikeFont class initialize | new* |
| StrikeFont class readAll: | new* |
| StrikeFont class readFrom: | new* |
| StrikeFont ascentForStdAsciiChars | new* |
| StrikeFont asTextStyle | new* |
| | |
| StrikeFont bottomLead: | new* |
| StrikeFont computeAscentDescentForStdAsciiChars | new* |
| StrikeFont descentForStdAsciiChars | new* |
| StrikeFont familySizeFace | modified* |
| StrikeFont glyphsSwitchCharacters | new* |
| | |
| StrikeFont initializeFrom: | new* |
| StrikeFont isFixedPitch | new* |
| StrikeFont leadInfo | new* |
| StrikeFont tightLeadInfo | new* |
| StrikeFont topLead | new* |
| StrikeFont type | new* |
| | |
| StrikeFont type: | new* |
| StrikeFont underLineInfo: | new* |
| StrikeFont writeOn: | new* |
| StrikeFont writeOnFile: | new* |
| StrikeFont xTableSwitchCharacters | new* |
| | |
| define StrikeFontManager | new* |
| | |
| StrikeFontManager class initialize | new* |
| | |
| StrikeFontManager at:ifAbsent: | new* |
| StrikeFontManager at:put: | new* |
| StrikeFontManager checkName: | new* |
| StrikeFontManager copy:name:emphasis: | new* |
| StrikeFontManager errorFontMissing: | new* |
| StrikeFontManager errorNameFormat: | new* |
| StrikeFontManager fontNames: | new* |
| StrikeFontManager install: | new* |
| StrikeFontManager install:ifAbsent: | new* |
| StrikeFontManager virtuallyAt: | new* |
| | |
| String class readDefinitionFrom:map: | new |
| String storeDefinitionOn:auxTable: | new |
| | |
| define StringHolderView | modified* |
| StringHolderView displayView: | modified* |
| StringHolderView editString: | modified* |

| | |
|---|---|
| StringHolderView initialize | modified* |
| StringHolderView textStyle | new* |
| StringHolderView textStyle: | new* |
| | |
| define StructOutputTable | new |
| | |
| StructOutputTable class new | new |
| StructOutputTable idOfElement:ifNew: | new |
| StructOutputTable if:isGlobal: | new |
| StructOutputTable new:globalDict: | new |
| | |
| Subtask class copyEnvironment | modified* |
| Subtask class currentEnvironment | modified* |
| Subtask class initializeEnvironment | modified* |
| | |
| define SwitchView | modified* |
| SwitchView displayView | new* |
| SwitchView initialize | modified* |
| SwitchView textStyle | new* |
| SwitchView textStyle: | new* |
| | |
| Symbol class readDefinitionFrom:map: | new |
| Symbol isUniqueValue | new |
| Symbol storeDefinitionOn:auxTable: | new |
| Symbol structureCopyWithDict: | new |
| | |
| SystemDictionary class readDefinitionFrom:map: | new |
| SystemDictionary appendChangesToSourceFileWithout: | new* |
| SystemDictionary copyright | modified* |
| SystemDictionary core | modified |
| SystemDictionary garbageCollect | new |
| SystemDictionary garbageCollect: | new |
| SystemDictionary getImageName | modified* |
| SystemDictionary install | modified* |
| SystemDictionary isUniqueValue | new |
| SystemDictionary lowSpaceNotificationLoop | modified |
| SystemDictionary resetSpaceLimits | modified |
| SystemDictionary shutdown | modified* |
| SystemDictionary snapshotAs:thenQuit: | modified* |
| SystemDictionary storeDefinitionOn:auxTable: | new |
| SystemDictionary structureCopyWithDict: | new |
| SystemDictionary version | modified* |
| | |
| define SystemTracer | new |
| | |
| SystemTracer class initialize | new |
| SystemTracer class write: | new |

| | |
|---|---|
| SystemTracer class writeClone | new |
| SystemTracer class writeCloneWithout: | new |
| SystemTracer allCallsOn:clampedBy: | new |
| SystemTracer clamp: | new |
| | |
| SystemTracer createHashEntryFor:using: | new |
| SystemTracer doitWithout: | new |
| SystemTracer gradeOf: | new |
| SystemTracer hasClamped: | new |
| SystemTracer hashEntryFor: | new |
| | |
| SystemTracer init: | new |
| SystemTracer initClampedClasses: | new |
| SystemTracer initDict | new |
| SystemTracer new:class:length:flags:grade:trace:write: | new |
| | |
| SystemTracer newHashForObject: | new |
| SystemTracer newSmallIntegerHash: | new |
| SystemTracer oopOf: | new |
| SystemTracer openPrivateFiles: | new |
| | |
| SystemTracer permutation:for: | new |
| SystemTracer permute:by: | new |
| SystemTracer preserve: | new |
| SystemTracer printDanglingRefs | new |
| SystemTracer restartCloneContext | new |
| | |
| SystemTracer resumptionContext | new |
| SystemTracer sizeInBytesOf: | new |
| SystemTracer sizeInOopsOf: | new |
| SystemTracer trace: | new |
| SystemTracer winnow: | new |
| | |
| SystemTracer writeBitField:on: | new |
| SystemTracer writeBytes: | new |
| SystemTracer writeChars: | new |
| SystemTracer writeClamped: | new |
| SystemTracer writeContext: | new |
| | |
| SystemTracer writeIdentityDictionary: | new |
| SystemTracer writeIdentityValues:from: | new |
| SystemTracer writeImage: | new |
| SystemTracer writeIndexablePointers: | new |
| SystemTracer writeLargePartOf:using: | new |
| | |
| SystemTracer writeMethodDictionary: | new |
| SystemTracer writeOopOf:on: | new |

| | |
|---|---|
| SystemTracer writePointerField:on: | new |
| SystemTracer writePointers: | new |
| | |
| SystemTracer writeProcess: | new |
| SystemTracer writeSet: | new |
| SystemTracer writeSpecial1 | new |
| SystemTracer writeSpecial2 | new |
| SystemTracer writeWords: | new |
| | |
| TekSystemCall class controlPty:command:mode: | new* |
| TekSystemCall class createPty | new* |
| TekSystemCall class execSystemUtility:withArgs: | modified* |
| TekSystemCall class getMachineType | new* |
| TekSystemCall class getRealMachineType | new* |
| TekSystemCall class fcntl:function: | new* |
| TekSystemCall class maxNameSize | modified* |
| TekSystemCall class rump:operation: | new* |
| TekSystemCall class setMachineType | new* |
| TekSystemCall class vfork | modified* |
| | |
| Text class initTextConstants | modified* |
| Text class initTextConstants2 | modified* |
| Text class initTextConstants3 | new* |
| | |
| TextCollector defaultContents | modified* |
| | |
| TextList class initialize | modified* |
| TextList class onList: | modified* |
| TextList class onList:style: | new* |
| | |
| TextList recomposeWithTextStyle: | new* |
| | |
| define TextStyle | modified* |
| TextStyle class default | new* |
| TextStyle class default: | new* |
| | |
| TextStyle alignment | new* |
| TextStyle alignment: | new* |
| TextStyle asListStyle | new* |
| TextStyle asMenuStyle | new* |
| | |
| TextStyle basalFontFor: | new* |
| TextStyle baseline | new* |
| TextStyle baseline: | new* |
| TextStyle baselineForLists | new* |
| | |
| TextStyle baselineForLists: | new* |

| | |
|---|---|
| TextStyle baselineForMenus | new* |
| TextStyle baselineForMenus: | new* |
| TextStyle boldFontFor: | new* |
| TextStyle boldItalicFontFor: | new* |
| TextStyle clearIndents | modified* |
| TextStyle defaultFont | new* |
| TextStyle descent | modified* |
| TextStyle firstIndent | new* |
| TextStyle firstIndent: | new* |
| TextStyle flushFonts | new* |
| TextStyle fontArray | new* |
| TextStyle fontArray: | new* |
| TextStyle fontAt: | modified* |
| TextStyle fontAt:put: | new* |
| TextStyle fontFor:emphasis: | new* |
| TextStyle fontFor:face: | modified* |
| TextStyle fontNamed: | modified* |
| TextStyle isFontBold: | modified* |
| TextStyle isFontBoldItalic: | modified* |
| TextStyle isFontItalic: | modified* |
| TextStyle isFontSubscripted: | modified* |
| TextStyle isFontSuperscripted: | modified* |
| TextStyle isFontUnderlined: | modified* |
| TextStyle italicFontFor: | new* |
| TextStyle leftMarginTabAt: | modified* |
| TextStyle lineGrid | new* |
| TextStyle lineGrid: | new* |
| TextStyle lineGridForLists | new* |
| TextStyle lineGridForLists: | new* |
| TextStyle lineGridForMenus | new* |
| TextStyle lineGridForMenus: | new* |
| TextStyle listStyleForFont:upperLead:lowerLead: | new* |
| TextStyle menuStyleForFont:upperLead:lowerLead: | new* |
| TextStyle nestingDepth | modified* |
| TextStyle newFontArray: | modified* |
| TextStyle nextTabXFrom:leftMargin:rightMargin: | modified* |
| TextStyle outputMedium | new* |
| TextStyle outputMedium: | modified* |
| TextStyle restIndent | new* |

| | |
|---|---|
| TextStyle restIndent: | new* |
| TextStyle rightIndent | new* |
| TextStyle rightIndent: | new* |
| TextStyle rightMarginTabAt: | modified* |
| | |
| TextStyle subscriptedFontFor: | modified* |
| TextStyle superscriptedFontFor: | modified* |
| TextStyle tabWidth | modified* |
| TextStyle underlinedFontFor: | modified* |
| TextStyle unSubscriptedFontFor: | modified* |
| | |
| TextStyle unSuperscriptedFontFor: | modified* |
| TextStyle unUnderlinedFontFor: | modified* |
| TextStyle upperLead:lowerLead: | new* |
| | |
| define TextStyleManager | new* |
| | |
| TextStyleManager class flushMenus | new* |
| TextStyleManager class initialize | new* |
| TextStyleManager class new: | new* |
| | |
| TextStyleManager at:put: | new* |
| TextStyleManager changeDefaultTextStyle | new* |
| TextStyleManager changeDefaultTextStyle: | new* |
| TextStyleManager fontNamesFromBaseNames: | new* |
| TextStyleManager fromUser | new* |
| | |
| TextStyleManager fromUser: | new* |
| TextStyleManager initializeMenus | new* |
| TextStyleManager removeAssociation:ifAbsent: | new* |
| TextStyleManager removeKey:ifAbsent: | new* |
| TextStyleManager styleName:baseNames: | new* |
| | |
| TextStyleManager styleName:baseNames:lead: | new* |
| TextStyleManager styleName:baseNames:upperLead:lowerLead: | new* |
| TextStyleManager styleName:fontNames: | new* |
| TextStyleManager styleName:fontNames:lead: | new* |
| TextStyleManager styleName:fontNames:upperLead:lowerLead: | new* |
| | |
| define TextView | modified* |
| TextView initialize | modified* |
| TextView textStyle | new* |
| TextView textStyle: | new* |
| | |
| True class readDefinitionFrom:map: | new |

| | |
|---|---|
| UndefinedObject class readDefinitionFrom:map: | new |
| UndefinedObject isUniqueValue | new |
| UndefinedObject structureCopyWithDict: | new |
| | |
| View computeInsetDisplayBox | modified* |
| View textStyle: | new* |
| WordArray class maxSize | modified |

## Removed Methods

| | |
|---|---|
| Behavior kindOfSubclass | remove* |
| DisplayScreen writeBitmapOn: | remove* |
| Object nextInstance | remove |
| PipeStream binary | remove* |
| PipeStream contentsOfEntireFile | remove* |
| PipeStream text | remove* |
| PopUpMenu labels:font:lines: | remove* |
| StandardSystemView displayBorder | remove* |
| StrikeFont ascent: | remove* |
| TextStyle flushFonts | remove* |

# SMALL OBJECT SPACE CHANGES

The following list gives the changes between the Tektronix Smalltalk-80 image, Version T2.1.2a and the present image, Version T2.1.3.

| | |
|---|---|
| Behavior removeSelectorSimply: | modified* |
| BitEditor class initialize | modified* |
| Browser removeClass | modified* |
| Browser renameClass | modified* |
| ChangeScanner scanClassExpression:do: | modified* |
| Character class readDefinitionFrom:map: | new* |
| Checker class classVariablesNotReferenced | new* |
| Checker class printClassVariablesNotReferencedOn: | new* |
| Class nonVariableSubclass:instanceVariableNames:c...: | new* |
| Class replaceNameWith: | new* |
| ClassDescription definition | modified* |
| ClassDescription kindOfSubclass | modified* |
| ClassDescription moveChangesToSources: | new* |
| Collection growSize | modified* |
| Collection maxSize | modified* |
| CompiledMethod class newBytes:flags:nTemps:nArgs:nSt... | new |
| CompiledMethod class newBytes:flags:nTemps:nStack:nLits: | new |
| CompiledMethod class nullSourceDescriptor | new |
| CompiledMethod class quickReturnPC | new |
| CompiledMethod nullSourceDescriptor | new |
| CompiledMethod openByteCodeStream | new |
| CompiledMethod setPrimitive: | new |
| CompiledMethod sourceDescriptor | new |
| CompiledMethod sourceDescriptor: | new |
| CompiledMethod sourceOffset | new |
| CompiledMethod trailerSize | new |
| Compiler evaluate:in:to:notifying:ifFail: | modified* |
| ContextPart at:put: | modified* |
| ContextPart class basicNew: | new* |
| ContextPart class new: | new* |
| ContextPart copy | new* |
| ControlManager discardCachedDisplayForms | new* |
| ControlManager restore | modified* |

| | |
|---|---|
| Cursor class currentCursor: | new* |
| Cursor centerCursorInViewport | modified* |
| | |
| Debugger bindingOf:forStore: | modified* |
| DisplayBitmap class basicNew: | new* |
| DisplayBitmap class new: | new* |
| | |
| DisplayScreen class currentDisplay: | modified* |
| DisplayScreen class displayExtent: | modified* |
| DisplayScreen resetFrom:extent: | modified* |
| DisplayScreen resetFrom:extent:offset: | modified* |
| DisplayScreen setDisplayStateFrom: | new* |
| DisplayScreen setMouseBounds: | modified* |
| DisplayScreen setMouseBoundsUpper:low... | modified* |
| DisplayScreen viewport | new* |
| DisplayScreen viewportCenter | new* |
| | |
| DisplayText class text: | modified* |
| DisplayText textStyle | new* |
| | |
| define DisplayTextView | modified* |
| DisplayTextView initialize | modified* |
| DisplayTextView textStyle | new* |
| DisplayTextView textStyle: | new* |
| | |
| Encoder noteSourceRange:forNode: | modified* |
| Encoder sourceMap | new* |
| Encoder sourceMap: | modified* |
| Encoder tempNames | modified* |
| | |
| FileDirectory fullName | modified* |
| | |
| FileList createDirectory | modified* |
| FileList createFile | modified* |
| FileList directoryMenu | modified* |
| FileList fileListMenu | modified* |
| FileList fileName: | new* |
| FileList newFileMenu | modified* |
| FileList resetFileMenu | modified* |
| | |
| FileStream class initialize | modified* |
| FileStream appendFileStream: | new* |
| FileStream binary | modified* |
| FileStream contentsOfEntireFile | modified* |
| FileStream nextPutAll: | modified* |
| | |
| FileStream nextPutAll:startingAt: | new* |

| | |
|---|---|
| FileStream nextPutAll:startingAt:to: | new* |
| FileStream padTo: | modified* |
| FileStream size | modified* |
| FileStream text | modified* |
| | |
| FillInTheBlank class example1 | modified* |
| FillInTheBlank class example2 | modified* |
| | |
| define Float | modified* |
| | |
| Float class initialize | modified* |
| Float class negativeInfinity | new* |
| Float class notANumber | new* |
| Float class notANumber: | new* |
| Float class positiveInfinity | new* |
| | |
| Float arcCos | modified* |
| Float arcSin | modified* |
| Float arcTan | modified* |
| | |
| Float exp | modified* |
| Float floorLog: | modified* |
| Float isInfinity | new* |
| Float isNAN | new* |
| Float isNegativeInfinity | new* |
| Float isNormal | new* |
| Float isPositiveInfinity | new* |
| | |
| Float ln | modified* |
| Float log | modified* |
| Float printOn: | modified* |
| Float printStructureOn: | new* |
| Float truncated | modified* |
| | |
| Form class readFormFile: | modified* |
| FormHolderView cancel | modified* |
| | |
| InputSensor currentCursor: | modified* |
| Inspector fieldList | modified* |
| | |
| MessageNode emitForEffect:on: | modified* |
| MessageNode emitForValue:on: | modified* |
| | |
| define ListView | modified* |
| ListView initialize | modified* |
| ListView list: | modified* |
| ListView selectionBox | modified* |

| | |
|---|---|
| ListView textStyle | new* |
| ListView textStyle: | new* |
| | |
| MethodContext adjustPCsForStructReading | new* |
| MethodContext adjustPCsForStructWriting | new* |
| MethodContext at:put: | modified* |
| MethodContext basicAt:put: | modified* |
| MethodContext setSender:receiver:method:arguments: | modified* |
| | |
| MethodDefinitionChange accept:notifying: | modified* |
| MethodDefinitionChange sourceFileAndPosition: | modified* |
| | |
| MethodNode generate: | modified* |
| MethodNode generateNoQuick | modified* |
| MethodNode sourceMap | modified* |
| | |
| NotifierView class openContext:label:contents: | modified* |
| NotifierView class openInterrupt:onProcess: | modified* |
| NotifierView textStyle: | new* |
| Number class readFrom: | modified* |
| | |
| Object shallowCopy | modified* |
| | |
| Paragraph recomposeWithTextStyle: | new* |
| ParagraphEditor class initialize | modified* |
| ParagraphEditor changeEmphasis: | modified* |
| ParagraphEditor emphasisDefault:keyedTo: | modified* |
| ParagraphEditor readKeyboard | modified* |
| | |
| Pen mandala:diameter: | modified* |
| PipeReadStream contentsOfEntireFile | modified* |
| Point negated | new* |
| | |
| define PopUpMenu | modified* |
| PopUpMenu class labels:lines: | modified* |
| PopUpMenu class labels:lines:alignment: | modified* |
| | |
| PopUpMenu labels:textStyle:lines: | new* |
| PopUpMenu markerTop: | modified |
| PopUpMenu rescan | modified |
| PopUpMenu reset | modified |
| | |
| PositionableStream through: | modified |
| PositionableStream upTo: | modified |
| | |
| define ProcessorScheduler | modified* |

| | |
|---|---|
| ProcessorScheduler class initialize | modified* |
| ProcessorScheduler absolutelyTheHighestPriority | new* |
| ProcessorScheduler executeWithoutPreemption: | new* |
| ProcessorScheduler highestPriority: | modified* |
| ProcessorScheduler resetPriorities | modified* |
| | |
| Project enter | modified* |
| ProjectController class initialize | new* |
| | |
| Rectangle class fromUser: | modified* |
| Rectangle negated | new* |
| ReturnNode emitForReturn:on: | modified* |
| ReturnNode emitForValue:on: | modified* |
| ReturnNode pc | modified* |
| | |
| Scanner scanFieldNames: | modified* |
| | |
| ScreenController forkOSshell | modified* |
| ScrollController canScroll | modified* |
| ScrollController canScrollDown | new* |
| ScrollController canScrollUp | new* |
| ScrollController controlInitialize | modified* |
| ScrollController moveMarker | modified* |
| ScrollController moveMarker: | modified* |
| ScrollController scrollDown | modified* |
| ScrollController scrollUp | modified* |
| SequenceableCollection hash | modified* |
| | |
| Set class maxSize | new* |
| Set class new | modified* |
| | |
| StandardSystemController class initialize | modified* |
| StandardSystemController textStyle | new* |
| StandardSystemController textStyle: | new* |
| | |
| define StandardSystemView | modified* |
| StandardSystemView getFrame | modified* |
| StandardSystemView initialize | modified* |
| StandardSystemView label: | modified* |
| StandardSystemView label:style: | new* |
| StandardSystemView resetLabel: | modified* |
| StandardSystemView resetLabel:style: | new* |
| StandardSystemView textStyle | new* |
| StandardSystemView textStyle: | new* |
| | |
| Stream do: | modified* |
| Stream nextPutAll:startingAt:to: | new* |

| | |
|---|---|
| define StrikeFont | modified* |
| StrikeFont class initialize | new* |
| StrikeFont class readAll: | new* |
| StrikeFont class readFrom: | new* |
| | |
| StrikeFont ascentForStdAsciiChars | new* |
| StrikeFont asTextStyle | new* |
| StrikeFont bottomLead: | new* |
| StrikeFont computeAscentDescentForStdAsciiChars | new* |
| StrikeFont descentForStdAsciiChars | new* |
| StrikeFont familySizeFace | modified* |
| | |
| StrikeFont glyphsSwitchCharacters | new* |
| StrikeFont initializeFrom: | new* |
| StrikeFont isFixedPitch | new* |
| StrikeFont leadInfo | new* |
| StrikeFont tightLeadInfo | new* |
| | |
| StrikeFont topLead | new* |
| StrikeFont type | new* |
| StrikeFont type: | new* |
| StrikeFont underLineInfo: | new* |
| StrikeFont writeOn: | new* |
| StrikeFont writeOnFile: | new* |
| StrikeFont xTableSwitchCharacters | new* |
| | |
| define StrikeFontManager | new* |
| | |
| StrikeFontManager class initialize | new* |
| | |
| StrikeFontManager at:ifAbsent: | new* |
| StrikeFontManager at:put: | new* |
| StrikeFontManager checkName: | new* |
| StrikeFontManager copy:name:emphasis: | new* |
| StrikeFontManager errorFontMissing: | new* |
| StrikeFontManager errorNameFormat: | new* |
| StrikeFontManager fontNames: | new* |
| StrikeFontManager install: | new* |
| StrikeFontManager install:ifAbsent: | new* |
| StrikeFontManager virtuallyAt: | new* |
| | |
| define StringHolderView | modified* |
| StringHolderView displayView: | modified* |
| StringHolderView editString: | modified* |
| StringHolderView initialize | modified* |
| StringHolderView textStyle | new* |

| | |
|---|---|
| StringHolderView textStyle: | new* |
| Subtask class copyEnvironment | modified* |
| Subtask class currentEnvironment | modified* |
| Subtask class initializeEnvironment | modified* |
| define SwitchView | modified* |
| SwitchView displayView | new* |
| SwitchView initialize | modified* |
| SwitchView textStyle | new* |
| SwitchView textStyle: | new* |
| SystemDictionary appendChangesToSourceFileWithout: | new* |
| SystemDictionary copyright | modified* |
| SystemDictionary getImageName | modified* |
| SystemDictionary install | modified* |
| SystemDictionary shutdown | modified* |
| SystemDictionary snapshotAs:thenQuit: | modified* |
| SystemDictionary version | modified* |
| TekSystemCall class controlPty:command:mode: | new* |
| TekSystemCall class createPty | new* |
| TekSystemCall class execSystemUtility:withArgs: | modified* |
| TekSystemCall class getMachineType | new* |
| TekSystemCall class getRealMachineType | new* |
| TekSystemCall class fcntl:function: | new* |
| TekSystemCall class maxNameSize | modified* |
| TekSystemCall class rump:operation: | new* |
| TekSystemCall class setMachineType | new* |
| TekSystemCall class vfork | modified* |
| Text class initTextConstants | modified* |
| Text class initTextConstants2 | modified* |
| Text class initTextConstants3 | new* |
| TextCollector defaultContents | modified* |
| TextList class initialize | modified* |
| TextList class onList: | modified* |
| TextList class onList:style: | new* |
| TextList recomposeWithTextStyle: | new* |
| define TextStyle | modified* |
| TextStyle class default | new* |
| TextStyle class default: | new* |

| | |
|---|---|
| TextStyle alignment | new* |
| TextStyle alignment: | new* |
| TextStyle asListStyle | new* |
| TextStyle asMenuStyle | new* |
| TextStyle basalFontFor: | new* |
| TextStyle baseline | new* |
| TextStyle baseline: | new* |
| TextStyle baselineForLists | new* |
| TextStyle baselineForLists: | new* |
| TextStyle baselineForMenus | new* |
| TextStyle baselineForMenus: | new* |
| TextStyle boldFontFor: | new* |
| TextStyle boldItalicFontFor: | new* |
| TextStyle clearIndents | modified* |
| TextStyle defaultFont | new* |
| TextStyle descent | modified* |
| TextStyle firstIndent | new* |
| TextStyle firstIndent: | new* |
| TextStyle flushFonts | new* |
| TextStyle fontArray | new* |
| TextStyle fontArray: | new* |
| TextStyle fontAt: | modified* |
| TextStyle fontAt:put: | new* |
| TextStyle fontFor:emphasis: | new* |
| TextStyle fontFor:face: | modified* |
| TextStyle fontNamed: | modified* |
| TextStyle isFontBold: | modified* |
| TextStyle isFontBoldItalic: | modified* |
| TextStyle isFontItalic: | modified* |
| TextStyle isFontSubscripted: | modified* |
| TextStyle isFontSuperscripted: | modified* |
| TextStyle isFontUnderlined: | modified* |
| TextStyle italicFontFor: | new* |
| TextStyle leftMarginTabAt: | modified* |
| TextStyle lineGrid | new* |
| TextStyle lineGrid: | new* |
| TextStyle lineGridForLists | new* |
| TextStyle lineGridForLists: | new* |
| TextStyle lineGridForMenus | new* |

| | |
|---|---|
| TextStyle lineGridForMenus: | new* |
| TextStyle listStyleForFont:upperLead:lowerLead: | new* |
| TextStyle menuStyleForFont:upperLead:lowerLead: | new* |
| TextStyle nestingDepth | modified* |
| TextStyle newFontArray: | modified* |
| | |
| TextStyle nextTabXFrom:leftMargin:rightMargin: | modified* |
| TextStyle outputMedium | new* |
| TextStyle outputMedium: | modified* |
| TextStyle restIndent | new* |
| TextStyle restIndent: | new* |
| | |
| TextStyle rightIndent | new* |
| TextStyle rightIndent: | new* |
| TextStyle rightMarginTabAt: | modified* |
| TextStyle subscriptedFontFor: | modified* |
| TextStyle superscriptedFontFor: | modified* |
| | |
| TextStyle tabWidth | modified* |
| TextStyle underlinedFontFor: | modified* |
| TextStyle unSubscriptedFontFor: | modified* |
| TextStyle unSuperscriptedFontFor: | modified* |
| | |
| TextStyle unUnderlinedFontFor: | modified* |
| TextStyle upperLead:lowerLead: | new* |
| | |
| define TextStyleManager | new* |
| | |
| TextStyleManager class flushMenus | new* |
| TextStyleManager class initialize | new* |
| TextStyleManager class new: | new* |
| | |
| TextStyleManager at:put: | new* |
| TextStyleManager changeDefaultTextStyle | new* |
| TextStyleManager changeDefaultTextStyle: | new* |
| TextStyleManager fontNamesFromBaseNames: | new* |
| TextStyleManager fromUser | new* |
| | |
| TextStyleManager fromUser: | new* |
| TextStyleManager initializeMenus | new* |
| TextStyleManager removeAssociation:ifAbsent: | new* |
| TextStyleManager removeKey:ifAbsent: | new* |
| TextStyleManager styleName:baseNames: | new* |
| | |
| TextStyleManager styleName:baseNames:lead: | new* |
| TextStyleManager styleName:baseNames:upperLead:lowerLead: | new* |
| TextStyleManager styleName:fontNames: | new* |

| | |
|---|---|
| TextStyleManager styleName:fontNames:lead: | new* |
| TextStyleManager styleName:fontNames:upperLead:lowerLead: | new* |
| | |
| define TextView | modified* |
| TextView initialize | modified* |
| TextView textStyle | new* |
| TextView textStyle: | new* |
| | |
| View computeInsetDisplayBox | modified* |
| View textStyle: | new* |

## Removed Methods

| | |
|---|---|
| Behavior kindOfSubclass | remove* |
| DisplayScreen writeBitmapOn: | remove* |
| PipeStream binary | remove* |
| PipeStream contentsOfEntireFile | remove* |
| PipeStream text | remove* |
| StandardSystemView displayBorder | remove* |
| PopUpMenu labels:font:lines: | remove* |
| StrikeFont ascent: | remove* |
| TextStyle flushFonts | remove* |

# *MANUAL CHANGE INFORMATION*

ARTIFICIAL INTELLIGENCE MACHINES DIVISION

**PRODUCT** 4404 SMALLTALK-80 INTRODUCTION USERS **PART NO** 070-5606-00

**PRODUCT GROUP** 07 **CHANGE NO** C35606 **DATE** MAY 1986

**This is an UPDATE package.**

1. Insert the attached UPDATE to the Smalltalk-80 LOS User Notes into the NOTES section of your manual.
2. Replace the title and manual revision pages at the front of your manual.
3. Keep this cover sheet in the CHANGE INFORMATION section at the back of this manual for a permanent record.

TEK USERS MANUAL

Part No. 070-5606-00
Product Group 07

# 4404

# ARTIFICIAL INTELLIGENCE SYSTEM

## INTRODUCTION TO THE SMALLTALK-80 SYSTEM

*Please Check at the Rear of this Manual for NOTES and CHANGE INFORMATION*

First Printing DEC 1984
Revised MAY 1986

Tektronix
COMMITTED TO EXCELLENCE

**PLEASE FORWARD ALL MAIL TO:**

**Artificial Intelligence Machines**
**Tektronix, Inc.**
**P.O. Box 1000 M.S. 60-405**
**Wilsonville, Oregon 97070**
**Attn: AIM Documentation**

# MANUAL REVISION STATUS

**PRODUCT: 4405 ARTIFICIAL INTELLIGENCE SYSTEM SMALLTALK-80 SYSTEM**

This manual supports the following versions of this product: Version T2.2.0

| REV DATE | DESCRIPTION |
|---|---|
| DEC 1984 | Original Issue |
| DEC 1985 | Subtask Management Notes |
| APR 1986 | LOS Users Notes |
| MAY 1986 | Update to LOS Users Notes |

# UPDATE

## About This Update

This Update expands the information about how to bring up Smalltalk on the 4405 AIS machine. If you have a 4405 *with just 1 megabyte of memory*, first refer to page 2 of the *Smalltalk-80 LOS User Notes*, then read this Update. If you have a 4405 with more than 1 megabyte of memory, the information in this Update does not affect you.

## Summary

On a *1 megabyte* 4405 machine, you should run only the Small Object Space (SOS) interpreter and images.

## Explanation

The 4405 and 4406 AIS machines are loaded with identical software at the factory. This is true whether you have ordered a standard machine with the minimum amount of memory or whether you have ordered a 4405 with one of the expanded memory options.

On the 4405, as on the 4406, you have the option of running the Small Object Space (SOS) interpreter or the Large Object Space (LOS) interpreter. *In a 4405 with just 1 megabyte of memory*, you should run only the SOS interpreter. You may run either the SOS or LOS interpreter on machines with more than 1 megabyte of memory.

## Invoking Smalltalk on a 1 Megabyte 4405

If you have a 1 megabyte 4405, the default is the SOS Smalltalk interpreter, invoked by any of the following:

*smalltalk*

*smalltalk <an_SOS_image_filename>*

*<an_SOS_image_filename>*

Note that by using the *filetype* command you can determine whether your image file is an SOS or LOS image file.

## The Demo Image

The demonstration image shipped with the 4405 and 4406 AIS machines is the same, and it is an LOS image. Thus, you should *not* run this image on a *1 megabyte* 4405.

# *MANUAL CHANGE INFORMATION*

ARTIFICIAL INTELLIGENCE MACHINES DIVISION

**PRODUCT** 4404 SMALLTALK-80 INTRODUCTION USERS **PART NO** 070-5606-00

**PRODUCT GROUP** 07 **CHANGE NO** C4-5606 **DATE** JUL 1986

**This is an Update package.**

1. Insert the attached Update into the NOTES section of your manual.
2. Replace the title and manual revision pages at the front of your manual.
3. Keep this cover sheet in the CHANGE INFORMATION section at the back of this manual for a permanent record.

TEK USERS MANUAL

Part No. 070-5606-00
Product Group 07

# 4404

# ARTIFICIAL INTELLIGENCE SYSTEM

## INTRODUCTION TO THE SMALLTALK-80 SYSTEM

*Please Check at the Rear of this Manual for NOTES and CHANGE INFORMATION*

First Printing DEC 1984
Revised JUL 1986

Tektronix
COMMITTED TO EXCELLENCE

**PLEASE FORWARD ALL MAIL TO:**

**Artificial Intelligence Machines**
**Tektronix, Inc.**
**P.O. Box 1000 M.S. 60-405**
**Wilsonville, Oregon 97070**
**Attn: AIM Documentation**

# MANUAL REVISION STATUS

**PRODUCT: 4405 ARTIFICIAL INTELLIGENCE SYSTEM SMALLTALK-80 SYSTEM**

This manual supports the following versions of this product: Version T2.2.0

| REV DATE | DESCRIPTION |
|---|---|
| DEC 1984 | Original Issue |
| DEC 1985 | Subtask Management Notes |
| APR 1986 | LOS Users Notes |
| MAY 1986 | Update to LOS Users Notes |
| JUL 1986 | Update |

# UPDATE

## 4405 SMALLTALK PROBLEM

Problem: At times, after using the middle button menu selection 'OS shell' the shell prompt (++) is not visible.

Cause: Smalltalk does not reset the machine's viewport to the upper left (0,0) position. The prompt exists, but not on the visible screen. Smalltalk also leaves joydisk panning enabled. Figure 1 shows the viewport (what you see on the 4405's screen) positioned away from the shell prompt.

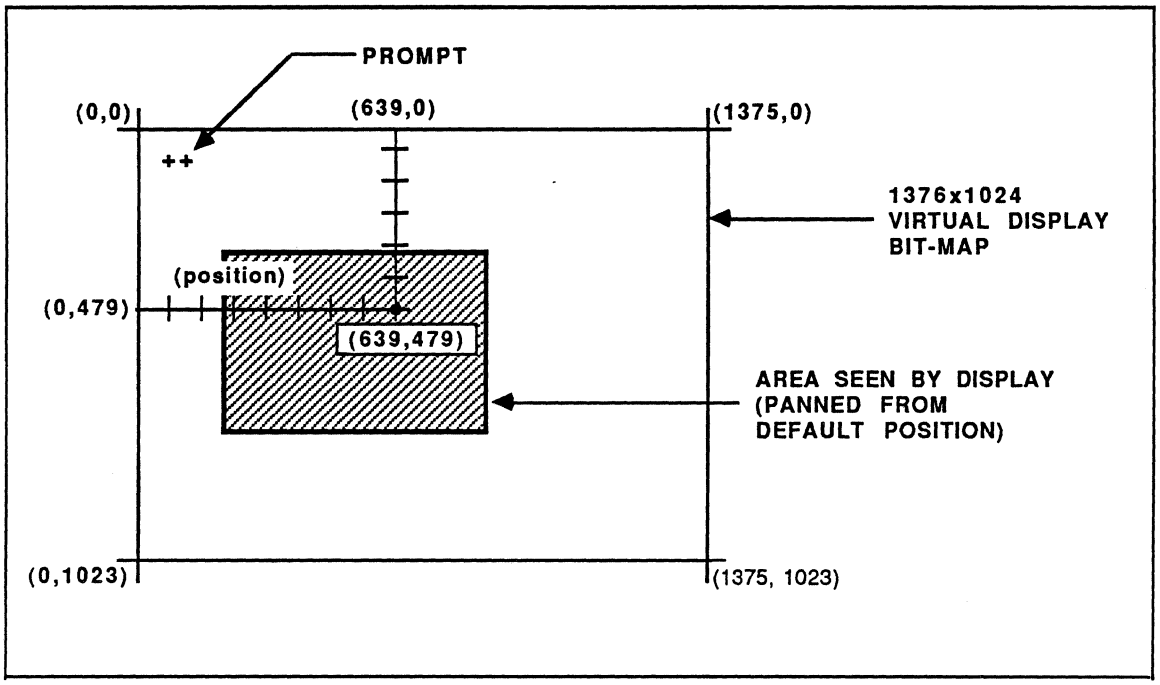


**Figure 1. Prompt Outside Viewport.**

Solution 1: Use the joydisk to pan to the upper left corner of the virtual display screen. If you then want to use the joydisk for history retrieval, disable joydisk panning with:

```
conset -diskpan
```

Solution 2 To prevent the problem in an image and subsequent snapshots of an image, modify the instance method 'forkOSshell' in class 'ScreenController' under category 'Interface-Support'. Select this method in the browser and insert the following two lines of code just *before* the line that begins with TekSystemCall.

——————————————code to insert——————————————

```
Display setViewportLocation: 0@0.

Display disableJoydiskPanning.
```

——————————————end of insert——————————————

Use the middle button menu selection and 'accept.' This image and subsequent snapshots will properly reset the viewport to (0,0). Figure 2 shows the system browser with the two new lines of code about to be accepted.

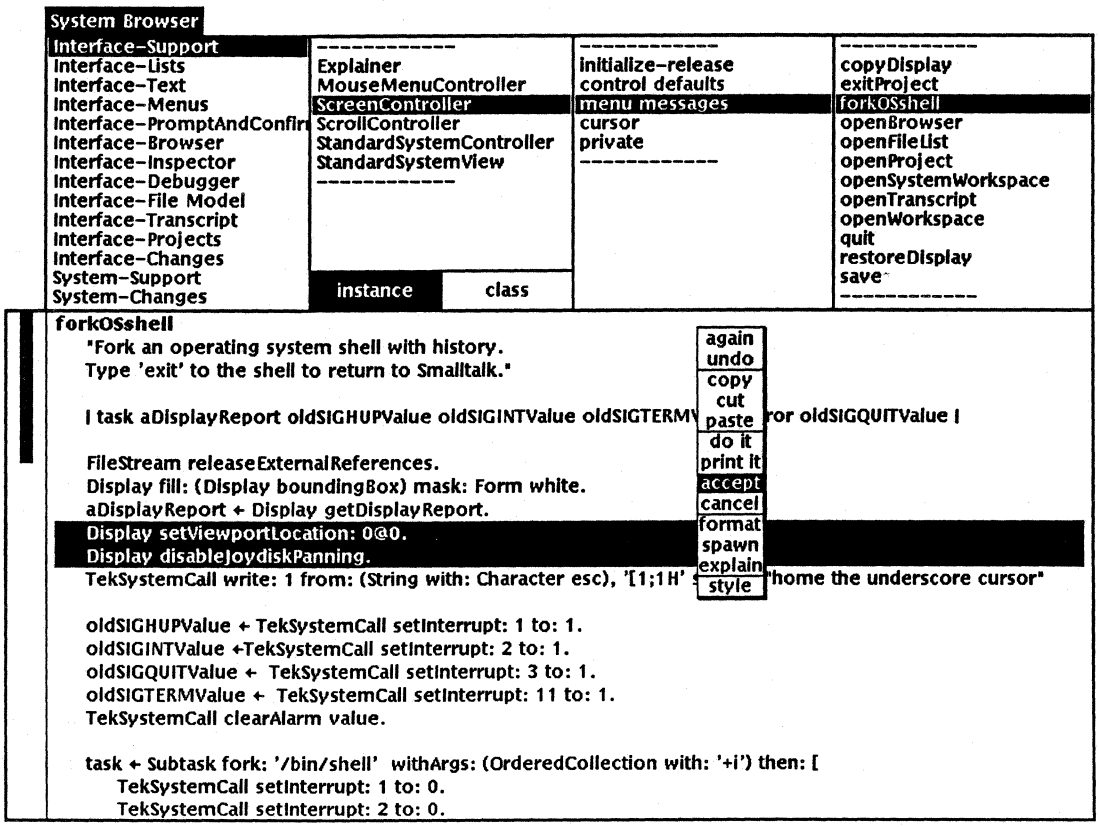


**Figure 2. Code in System Browser.**

# *MANUAL CHANGE INFORMATION*

ARTIFICIAL INTELLIGENCE MACHINES DIVISION

**PRODUCT** 4404 SMALLTALK-80 INTRO. USERS **PART NO** 070-5606-00

**PRODUCT GROUP** 07 **CHANGE NO** C6-5606 **DATE** JAN 1987

This is a CHANGE and ADDITION package.

1. Insert the attached 4404 SOS Release Notes into the Notes section of your manual.
2. Replace the title and manual revision pages at the front of your manual.
3. Keep this cover sheet in the CHANGE INFORMATION section at the back of this manual for a permanent record.

TEK USERS MANUAL

Part No. 070-5606-00
Product Group 07

# 4404
# ARTIFICIAL INTELLIGENCE SYSTEM
## INTRODUCTION TO THE SMALLTALK-80 SYSTEM

*Please Check at the Rear of this Manual for NOTES and CHANGE INFORMATION*

First Printing DEC 1984
Revised JAN 1987

Tektronix
COMMITTED TO EXCELLENCE

**PLEASE FORWARD ALL MAIL TO:**

**Artificial Intelligence Machines**
**Tektronix, Inc.**
**P.O. Box 1000 M.S. 60-405**
**Wilsonville, Oregon 97070**
**Attn: AIM Documentation**

# MANUAL REVISION STATUS

**PRODUCT: 4400 SERIES AIM SMALLTALK-80 SYSTEM**

This manual supports the following versions of this product: LOS Version T2.2.0c and SOS Version T2.1.3b

| REV DATE | DESCRIPTION |
|---|---|
| DEC 1984 | Original Issue |
| DEC 1985 | Subtask Management Notes |
| APR 1986 | LOS Users Notes |
| MAY 1986 | Update to LOS Users Notes |
| JUL 1986 | Update |
| JAN 1987 | 4404 SOS Release Notes |
| JAN 1987 | 4405/06 Release Notes |

# Table of Contents

# Figures

# 4404 SOS Release Notes

These notes document the Tektronix SOS Smalltalk-80[1] system Version T2.1.3b for the 4404 Tektronix Artificial Intelligence Machine.

## ABOUT THESE NOTES

This document consists of the notes themselves and two appendices. The contents are:

- The notes themselves document the differences, additions, changes, etc., since the last release of the Tektronix SOS Smalltalk-80 system.
- Appendix A *Smalltalk-80 Version T2.1.3b Files.* For your convenience, you will find a list of all files associated with the SOS Version T2.1.3b release.
- Appendix B *SOS Image Version Changes.* For your convenience, you will find a list of all changes to classes and methods in this latest (T2.1.3b) release of the SOS image.

## SOS AND LOS INTERPRETERS

Tektronix has created two distinct interpreters for the Smalltalk-80 system. The original interpreter was developed along the lines of the interpreter specified in the Addison-Wesley Smalltalk book by Adele Goldberg and David Robson. This interpreter, called the SOS interpreter, allows you to develop small- to medium-sized applications. The development of more powerful hardware – the 4405 and 4406 AIM systems – led to the creation of a new interpreter. This new interpreter is called the LOS (Large Object Space) interpreter. The LOS interpreter allows you to have an effectively unlimited number of objects, which means that very large size applications are now possible to develop using the LOS interpreter.

### The SOS Interpreter

The Tektronix SOS (Small Object Space) interpreter is essentially the interpreter specified in the Addison-Wesley "blue book". (Goldberg, Adele and David Robson. Smalltalk-80: The Language and its Implementation. Addison-Wesley, 1983.) See that book for more information about the interpreter.

The SOS interpreter runs on the 4404, 4405, and 4406 AIS machines. However, it is strongly recommended that you use the LOS interpreter on the 4405 and 4406 machines since the LOS interpreter has more features and places fewer contraints on your applications than the SOS interpreter. You should even consider converting your work developed in an SOS image on a 4404 to an LOS image on a 4405 or 4406 if you have purchased one of these faster machines.

---

1. Smalltalk-80 is a Trademark of Xerox Corporation.

Note that the LOS interpreter takes advantage of hardware features on the 4405 and 4406 and, thus, is not available for the 4404 machine.

# INVOKING SMALLTALK

The standard 4404 AIM system has resident on it one Smalltalk system, consisting of an interpreter and a virtual image: the SOS (Small Object Space) Smalltalk-80 system. To bring up the system, type:

*smalltalk*

at the system prompt. This brings up the SOS interpreter, which then loads the SOS virtual image file.

# PRIMITIVE METHODS

Some Smalltalk methods are implemented by making machine language calls directly. These methods are called primitive methods.

## String Comparison Primitive

One primitive has been added to the SOS interpreter:

148 string = – Answers true if the receiver and argument contain the same ASCII characters. Answers false if not. Fails if the class of the argument is different from the class of the receiver.

# IMAGE MODIFICATIONS

The SOS image has a number of modifications. Here are some highlights.

- Many View subclasses (ListView, StandardSystemView, StringHolderView, and TextView, for example) have been redefined for augmented access to fonts. PopUpMenu, StrikeFont, and TextStyle classes also changed and two new classes, StrikeFontManager and TextStyleManager, have been added. See *Enhanced Font Support* for more details.
- Additional protocol has been added to TekSystemCall related to pseudo-ttys and access to the machine name.

# USER INTERFACE CHANGES

There are two changes here. New window framing gives you a bit more control of window placement than before, and a new blue button menu item, style, allows you to easily switch between text styles in a window.

## New Window Framing

Windows now frame by letting you switch between moving the top-left and bottom-right corners until you get them placed exactly how you want them.

When you want to frame a window via the normal user interface, for example, the "top-left" cursor appears, which you may move around on the screen. When it is approximately in the right position, you press the left mouse button, causing the "bottom-right" cursor to appear. Once you have located the bottom-right corner, you have two options. The first is to remove your finger from the mouse button completely; this has the effect of selecting the rectangle just framed. The second option is for you to lift your finger from the mouse for just an instant and to immediately press it again. This has the effect of moving the cursor back to the top-left corner of the rectangle, allowing you to adjust your original placement of that corner. When you are finished with the top-left corner, you again may move back to the bottom-right corner in the same manner, etc.

The determination of whether you have "quick-clicked" or not is made by an instance of class Delay, which is created in the method getFrame. There is a constant in this routine that specifies the time in milliseconds to wait. This constant is currently set at 250 (or 1/4 of a second); you can set it to another value by modifying the StandardSystemView getframe method.

## Blue Button Menu

The right button menu of StandardSystemViews has a new item – style. This allows a change of text style for a particular window, including its subviews. Available text styles are determined by the contents of StyleManager. See the System Workspace for an example of how to add text styles to your image.

# ENHANCED FONT SUPPORT

Smalltalk has new default fonts, a larger variety of fonts, and augmented access to the fonts. Available fonts range from very small to very large, serif and sans serif, and proportional and monospaced fonts. These fonts have an additional face – bold italic. Protocol for adding fonts and text styles to an image has been defined.

# Fonts

Three properties (family, face, and size) are commonly associated with a font. Family is the intrinsic property. Families are named and frequently protected by copyright. Examples include "Helvetica" and "Times Roman". Face is the emphatic property. Examples include Basal (no emphasis), Bold, Italic, BoldItalic, and Underlined. Size is the dimensional property. It is typically specified by the height of capital "A" in points (72nds of one inch), although such a measure is more meaningful on paper than on a display.

A Smalltalk-80 font is an instance of class StrikeFont, which represents a single combination of family, face, and size values with a bitmap for each character. In some cases, a face (other than Basal) is synthesized by bitmap manipulation of the Basal face. Examples include copying and offsetting (Bold), shearing (Italic), and underlining.

# Available Fonts

This product release includes the families: Pellucida[2] Sans-Serif, Pellucida Serif, Pellucida Typewriter, Xerox Sans-Serif, and Xerox Serif. The Pellucida Sans-Serif, Pellucida Serif, and Xerox families are proportionally spaced (individual characters within the same font have varying widths); the Pellucida Typewriter family is monospaced (individual characters within the same font have the same width). The Pellucida families are new to this product release; the Xerox families are the standard Smalltalk-80 Version 2 fonts.

Fonts are stored using a standard file format within the directory */fonts*. The name of a file in this directory should be the name of the font it holds suffixed with *.font*.

The name of a StrikeFont is a String with three components (family, size, and face) and no embedded spaces. The family component is the family name with spaces removed; the size component is the printString of the numeric size; and the face component is a String of length zero, one, or two encoding the emphasis. The supported face codes are "" (Basal), "B" (Bold), "I" (Italic), "X" (BoldItalic), "U" (Basal Underlined), "BU" (Bold Underlined), "IU" (Italic Underlined), and "XU" (BoldItalic Underlined). Examples of names include "PellucidaSans-Serif8", "XeroxSerif12I", and "Typewriter18BU".

The Pellucida Sans-Serif and Serif fonts are available in four non-synthetic faces (Basal, Bold, Italic, and BoldItalic) and seven sizes (8, 10, 12, 14, 18, 24, and 36 point); see Figure 1-3, *Tektronix Proportional Fonts (PellucidaSerif and PelluciaSans-Serif)*, for the character set ordering. The Pellucida Typewriter fonts are available in two non-synthetic faces (Basal and Bold) and four sizes (10, 12, 16, and 18 point); see Figure 1-4, *Tektronix Monospaced Fonts (Pellucia Typewriter) Part 1* and Figure 1-5, *Tektronix Monospace Fonts (Pellucida Typewriter) Part 2*, for the character set ordering. The Xerox fonts are available in three non-synthetic faces (Basal, Bold, and Italic) and two sizes (10 and 12 point), although the Sans-Serif Italic 10 point font is synthetic.

2. Pellucida is a registered trademark of Bigelow and Holmes.

# Interpreting Font Tables

A few notes on interpreting the font tables will be helpful in constructing an application. The spaces in the table that are blank do not have a printing character for the corresponding character code. The characters for ASCII 32 through ASCII 127 are present in both the monospaced and proportional fonts. The proportional fonts contain additional characters in ASCII 1 through ASCII 31. Many of these characters are compatible with those originally supplied by Xerox in the standard Smalltalk-80 Version 2 image.

"m space" is a blank character which is the height and width of the letter m. "n space" is a blank character which is the height and width of the letter n. "em" and "en" are dashes the width of the character "m" and "n", respectively.

# Reading and Writing

Smalltalk StrikeFont class has methods for reading and writing Tektronix font files. Note that whenever Smalltalk reads a Tektronix font file, it switches the character position of the uparrow character (↑) and left arrow (←) with the caret (^) and underscore (_) characters. Thus, if you ask, for instance, the character ↑ what its asciiValue is, you get 94.

The method to write a StrikeFont takes care to switch the positions of the ↑, ←, ^, and _ characters if the type of the strike font is either 1 (Tektronix monospaced) or 2 (Tektronix proportionally spaced). This ensures that the proportional or monospaced fonts written by Smalltalk have consistent character ordering.

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $10_{0_0}$ | $10_{0_1}$ | $10_{1_0}$ | $10_{1_1}$ | $11_{0_0}$ | $11_{0_1}$ | $11_{1_0}$ | $11_{1_1}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | ~ | space | 0 | @ | P | ‘ | p |
| 0 0 0 1 | ˇ | ffi | ! | 1 | A | Q | a | q |
| 0 0 1 0 | ¿ | ffl | ” | 2 | B | R | b | r |
| 0 0 1 1 | Ç | em | # | 3 | C | S | c | s |
| 0 1 0 0 | ¨ | fi | $ | 4 | D | T | d | t |
| 0 1 0 1 | ` | fl | % | 5 | E | U | e | u |
| 0 1 1 0 | ff | en | & | 6 | F | V | f | v |
| 0 1 1 1 | ‘ | ˘ | ’ | 7 | G | W | g | w |
| 1 0 0 0 | ¡ | — | ( | 8 | H | X | h | x |
| 1 0 0 1 | | n space | ) | 9 | I | Y | i | y |
| 1 0 1 0 | | ∞ | * | : | J | Z | j | z |
| 1 0 1 1 | ´ | ↑ | + | ; | K | [ | k | { |
| 1 1 0 0 | | ← | , | < | L | \ | \| | \| |
| 1 1 0 1 | | . | - | = | M | ] | m | } |
| 1 1 1 0 | ¯ | ~ | . | > | N | ^ | n | ~ |
| 1 1 1 1 | m space | ° | / | ? | O | _ | o | ▮ |

**Figure 1-1. Tektronix Proportional Fonts (PellucidaSerif and PellucidaSans-Serif).**

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $^{1}0_{0}0_{0}$ | $^{1}0_{0}0_{1}$ | $^{1}0_{1}0_{0}$ | $^{1}0_{1}0_{1}$ | $^{1}1_{0}0_{0}$ | $^{1}1_{0}0_{1}$ | $^{1}1_{1}0_{0}$ | $^{1}1_{1}0_{1}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | | | space | 0 | @ | P | ‘ | p |
| 0 0 0 1 | | | ! | 1 | A | Q | a | q |
| 0 0 1 0 | | | ” | 2 | B | R | b | r |
| 0 0 1 1 | | | # | 3 | C | S | c | s |
| 0 1 0 0 | | | $ | 4 | D | T | d | t |
| 0 1 0 1 | | | % | 5 | E | U | e | u |
| 0 1 1 0 | | | & | 6 | F | V | f | v |
| 0 1 1 1 | | | ’ | 7 | G | W | g | w |
| 1 0 0 0 | | | ( | 8 | H | X | h | x |
| 1 0 0 1 | | n space | ) | 9 | I | Y | i | y |
| 1 0 1 0 | | ↑ | * | : | J | Z | j | z |
| 1 0 1 1 | | ← | + | ; | K | [ | k | { |
| 1 1 0 0 | | | , | < | L | \ | \| | \| |
| 1 1 0 1 | | | - | = | M | ] | m | } |
| 1 1 1 0 | | | . | > | N | ^ | n | ~ |
| 1 1 1 1 | m space | | / | ? | O | _ | o | ■ |

**Figure 1-2. Tektronix Monospaced Fonts (Pellucida Typewriter) Part 1.**

| B8 B7 B6 B5 BITS B4 B3 B2 B1 | $10_{0_0}$ | $10_{0_1}$ | $10_{1_0}$ | $10_{1_1}$ | $11_{0_0}$ | $11_{0_1}$ | $11_{1_0}$ | $11_{1_1}$ |
|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | NU 128 | DL 144 | SP 160 | 0 176 | – 192 | Ñ 208 | ◆ 224 | 240 |
| 0 0 0 1 | SH 129 | D1 145 | Ä 161 | 1 177 | ¢ 193 | ñ 209 | ■ 225 | 241 |
| 0 0 1 0 | SX 130 | D2 146 | ä 162 | 2 178 | ¡ 194 | ¿ 210 | HT 226 | 242 |
| 0 0 1 1 | EX 131 | D3 147 | Å 163 | 3 179 | † 195 | ¡ 211 | FF 227 | 243 |
| 0 1 0 0 | ET 132 | DA 148 | å 164 | 4 180 | □ 196 | α 212 | CR 228 | 244 |
| 0 1 0 1 | EQ 133 | NK 149 | Æ 165 | 5 181 | ■ 197 | σ 213 | LF 229 | 245 |
| 0 1 1 0 | AK 134 | SY 150 | æ 166 | 6 182 | ● 198 | τ 214 | o 230 | 246 |
| 0 1 1 1 | BL 135 | EB 151 | à 167 | 7 183 | Δ 199 | ρ 215 | ± 231 | 247 |
| 1 0 0 0 | BS 136 | CN 152 | Ç 168 | 8 184 | δ 200 | μ 216 | NL 232 | 248 |
| 1 0 0 1 | HT 137 | EM 153 | é 169 | 9 185 | λ 201 | Σ 217 | VT 233 | ≤ 249 |
| 1 0 1 0 | LF 138 | SB 154 | è 170 | ù 186 | | Ω 218 | 234 | ≥ 250 |
| 1 0 1 1 | VT 139 | EC 155 | Ö 171 | β 187 | | ∫ 219 | 235 | π 251 |
| 1 1 0 0 | FF 140 | FS 156 | ö 172 | ⊙ 188 | | 220 | 236 | ≠ 252 |
| 1 1 0 1 | CR 141 | GS 157 | ø 173 | ¤ 189 | | ÷ 221 | 237 | £ 253 |
| 1 1 1 0 | SO 142 | RS 158 | Ü 174 | § 190 | ¬ 206 | ≈ 222 | 238 | ■ 254 |
| 1 1 1 1 | SI 143 | US 159 | ü 175 | ▪▪ 191 | ∝ 207 | ⌐ 223 | 239 | DT 255 |

**Figure 1-3. Tektronix Monospaced Fonts (Pellucida Typewriter) Part 2.**

The new class StrikeFontManager is a subclass of Dictionary and stores Associations between String names and StrikeFonts. A single instance of StrikeFontManager is known as the global FontManager. Particularly useful messages to this object include:

```
FontManager inspect.
FontManager fontNames: anArray.
```

The inspect method opens a DictionaryInspector on FontManager.

The fontNames: method returns an Array of StrikeFonts corresponding to anArray of String names. It attempts to load a font from the system font directory (*/fonts*) if that font is not already resident (contained within FontManager). The name of a file in this directory should be the name of the font it holds suffixed with ".font". The method further attempts to synthesize a font if it is not already resident and cannot be located within the system font directory.

# Text Styles

Most text processing in Smalltalk-80 is performed not with instances of class StrikeFont in isolation but rather with instances of class TextStyle, whose properties include:

- fontArray (an Array of StrikeFonts)
- lineGrid (distance from top of text line to top of next text line)
- baseline (distance from top of text line to base of capital letters)
- additional lineGrids and baselines for lists and menus
- alignment code (flush left, flush right, centered, justified)
- indentation and tab stop parameters

These properties of a TextStyle, as its name implies, are mostly a matter of personal style and system convention. The fonts are usually members of a single family (although the system default, described below, violates this rule for historical reasons) in one or two sizes and several faces. The lineGrid (termed "leading" by typographers) is usually the height of the tallest font in the style plus a certain amount of additional white space. The baseline is shared by all of the fonts in the style so that the bases of their capital letters are aligned. Subscripts and superscripts, of course, would violate this rule, but they are not supported in this product release (although rudimentary capabilities do exist within classes StrikeFont, TextStyle, and DisplayScanner). Flush left alignment has historically been the default in Smalltalk-80, but other possibilities are certainly available.

Obvious uses of TextStyles include class ParagraphEditor and its subclasses (in Workspaces, System Transcripts, Projects, and the bottom panes of System Browsers, File Lists, and ChangeListViews). Less obvious uses include lists, menus, and title tabs of StandardSystemViews. Even less obvious uses include the String messages asParagraph and asDisplayText. This broad variety of uses prompts some common questions:

- What is the system default style?
- Can additional styles coexist?
- If so, how are they created and catalogued?

- How can the system default style be changed?
- How can the style of a view or subview be changed?

These questions are addressed in the following paragraphs.

The system default style is mentioned in chapter three of the Goldberg ("orange") book. It contains twenty-four fonts (two families, two sizes, and six faces) ordered as follows:

- Sans-Serif 10 (Basal, Bold, Italic)
- Serif 12 (Basal, Bold, Italic)
- Serif 10 (Basal, Bold, Italic)
- Sans-Serif 12 (Basal, Bold, Italic)
- all of the above repeated but Underlined

The original Smalltalk-80 system default style used the Xerox sans serif and serif font families. This product release maintains other characteristics of that style (including the unusual mixing of sans serif and serif families) but uses the Pellucida families in the default text style.

The new class TextStyleManager is a subclass of Dictionary and stores Associations between String names and TextStyles. A single instance of TextStyleManager is known as the global StyleManager. Particularly useful messages to this object include:

```
StyleManager inspect.
StyleManager
  styleName: aString
  fontNames: anArrayOfStrings
  lead: anInteger.
StyleManager
  styleName: aString
  baseNames: anArrayOfStrings
  lead: anInteger.
```

The inspect message opens a DictionaryInspector on StyleManager.

The styleName:fontNames:lead: and styleName:baseNames:lead: methods return a new style named aString. The fontNames: version accepts font names with arbitrary face codes in anArrayOfStrings whereas the baseNames: version accepts only Basal font names and imposes the following order on the fonts:

- (anArrayOfStrings at: 1) Basal
- (anArrayOfStrings at: 1) Bold
- (anArrayOfStrings at: 1) Italic
- (anArrayOfStrings at: 1) BoldItalic
- similar sequence(s) for other element(s) of anArrayOfStrings (if any)
- all of the above repeated but Underlined

The actual font array is obtained from FontManager via the fontNames: message thereby invoking the font loading and synthesizing mechanisms discussed above. The lead: parameter is the amount of additional white space to add to the height of the tallest font to obtain the lineGrid for the style. Both methods also install the new style in StyleManager for future reference.

Thus the expression that created the system default style is:

```
StyleManager
  styleName: 'Pellucida Default 10 and 12'
  fontNames: #(
    'PellucidaSans-Serif10'
    'PellucidaSans-Serif10B'
    'PellucidaSans-Serif10I'
    'PellucidaSerif12'
    'PellucidaSerif12B'
    'PellucidaSerif12I'
    'PellucidaSerif10'
    'PellucidaSerif10B'
    'PellucidaSerif10I'
    'PellucidaSans-Serif12'
    'PellucidaSans-Serif12B'
    'PellucidaSans-Serif12I'

    'PellucidaSerif10U'
    'PellucidaSerif10BU'
    'PellucidaSerif10IU'
    'PellucidaSerif12U'
    'PellucidaSerif12BU'
    'PellucidaSerif12IU'
    'PellucidaSans-Serif10U'
    'PellucidaSans-Serif10BU'
    'PellucidaSans-Serif10IU'
    'PellucidaSans-Serif12U'
    'PellucidaSans-Serif12BU'
    'PellucidaSans-Serif12IU')
  lead: 3.
```

A similar expression using the base name technique is found in the System Workspace:

```
StyleManager
  styleName: 'Pellucida Sans-Serif 12 and 14'
  baseNames: #('PellucidaSans-Serif12' 'PellucidaSans-Serif14')
  lead: 3.
```

These expressions illustrate two style conventions. The first suggests font family and size in the style name. Mixing sans serif and serif families in one style, preferably with the font ordering convention described in the Goldberg book, is connoted by the common font family name prefix (assuming there is one!) concatenated with the word "Default". Thus the original Smalltalk-80 style name would be "Xerox Default 10 and 12". Note that embedded spaces are encouraged in style names (unlike font names, which must be storable in the filing system). The second convention is the use of three additional pixels of leading in styles mixing two near sizes of fonts. Most text in the context of the style is expected to be in one of the smaller fonts.

Expressions similar to these can be found in files in the Smalltalk text style directory (*/smalltalk/textStyles*). These files store not styles but rather expressions that create styles; this distinction is suggested by the file suffix ".ws" (an abbreviation for ".workspace"). These files can be filed in if wholesale style acquisition is desired, or specific expressions can be executed to acquire specific styles. The System Workspace holds an expression that references this

directory to discard all existing fonts and styles, read in small fonts, and create a new default style (useful on systems with a small screen):

```
Compiler evaluate:
  ((Disk file: '/smalltalk/textStyles/pruneToPellucidaDefault08and10.ws')
    contentsOfEntireFile).
```

The StyleManager maintains the mapping from style names to styles for future reference when it is desired to change the system default style or the style of a view or subview. The expression:

```
StyleManager changeDefaultTextStyle.
```

pops up a menu of resident styles, waits for a style to be selected with any mouse button (or aborts if the button is released outside the menu), and then changes the default style to the selected style. This also rebuilds system menus and recomposes text in scheduled views and subviews in the current project. This capability can be added as the "style" entry of the middle button menu of class ScreenController by filing in:

*/smalltalk/fileIn/addTextStyleToSystemMenu.st*

A subset of this capability (propagate the selected style to the title tab and subviews of the current view) is available as the "style" entry of the right button menu of scheduled controllers. See *User Interface Changes.*

An even smaller subset of this capability (propagate the selected style only to the subview) can be added as the "style" entry of the middle button menu of class ParagraphEditor and several of its subclasses by filing in:

*/smalltalk/fileIn/addTextStyleToYellowButton.st*

The methods at:, at:put:, and removeKey: are useful for more primitive manipulation of StyleManager; the last two automatically update the menu of resident styles. Note that changing a style in StyleManager by itself usually has no effect on any text since styles are typically copied before use. An experimental style can be tested by adding it to StyleManager and then selecting it with the appropriate menu button.

The method initializeMenus rebuilds system menus. It references several lists that should be extended for applications with private menus.

# Miscellaneous

Class StrikeFont has new instance variables ascentForStdAsciiChars and descentForStdAsciiChars. These maintain the envelope of characters space (Ascii Decimal Equivalent 032) through tilde (ADE 126) for use by class TextStyle to compute styles for lists and menus (see below). A simple TextStyle can be constructed by sending asTextStyle to an instance of StrikeFont. Finally, the metaclass understands readAll: and readFrom:; the latter is used by class StrikeFontManager to load fonts from the filing system.

Class TextStyle has new instance variables lineGridForLists, baselineForLists, lineGridForMenus, and baselineForMenus. These support conversion of the style for lists and menus with the methods asListStyle and asMenuStyle. The method flushFonts has been removed.

Class PopUpMenu (and hence class ActionMenu) replaces instance variable font with textStyle. The private method labels:font:lines: has been replaced by labels:textStyle:lines:. The metaclass understands labels:lines:alignment: so that non-centered alignments may be easily specified.

Within class ParagraphEditor, the control-x key formerly de-emphasized the current selection. Now control-x switches to a BoldItalic font (if possible), control-X switches to a non-BoldItalic font, and control-e de-emphasizes.

A new instance variable textStyle has been added to classes DisplayTextView, ListView, StandardSystemView, StringHolderView, SwitchView, and TextView. A method recomposeWithTextStyle: has been added to classes Paragraph and TextList. Class FileList has been modified not to cache menus in instance variables. Within pool dictionary TextConstants, DefaultLineGrid and DefaultBaseline have been removed; CtrlX and Ctrle have been added; and Ctrlx has been changed.

# STORING AND RETRIEVING OBJECTS ON FILES

You can include in your SOS image a mechanism for storing and retrieving object representations (including objects with circularities) on a file (or other character stream). This mechanism has two advantages over the original Smalltalk storeOn: mechanism. First, storeOn: does not work for objects that contain circularities; second, storeOn:'s output is meant to be read in by the compiler which limits the number of literals in an object to 64. Thus, storeOn: will not correctly handle all object structures.

The */smalltalk/conversion* directory contains a file, with this reading and writing mechanism, called *structSOSPackage.st.*. Incorporate this package into your SOS image by filing it in and using the methods below. This package also contains a copying mechanism discussed in the next section. This package may be used to transfer structures between LOS (Large Object Space) and SOS images.

## Using the Reading and Writing Mechanism

Four *visible* messages are defined to provide the writing or reading of objects to or from files or character-streams.

## To Write Structures:

someObject storeStructureOn: aStream.

Stores an object representation on a character stream aStream.

someObject storeStructureOnFile: aString.

Stores an object representation on a file named aString.

## To Read Structures:

Object readStructureFrom: aStream.

Answers an object defined by stream aStream.

Object readStructureFromFile: aString.

Answers an object defined on a file named aString.

These programs should allow object representations to be written or read to or from string format.

# Implementation Details

This mechanism maps objects based on == (equality). If an object has a circular structure when written out, it will be circular when read back in. Similarly, acyclic structures are read back in as acyclic structures. There are a few cautions, however:

1. There are some objects, such as processes, that may cause unexpected behavior if an attempt is made to write them out, or particularly to read back in. Contexts are treated specially in that the sender is always written out as nil. CompiledMethods are written out in a special format, which is compatible with both SOS and LOS images. Also be aware that the receiver of a MethodContext in which the block context was created is also copied as part of the definition of the MethodContext.
2. Smalltalk treats certain objects in a special way, guaranteeing their uniqueness. A new selector, isUniqueValue, has been defined that returns a boolean value, stating whether the object has this property. Such classes include UndefinedObject, Boolean, Symbol, SmallInteger and Character. Objects in these classes are mapped to the corresponding object in the target image. Floating point numbers are written out to 9 digits of accuracy. If more (or less) accuracy is desired, it is necessary to modify the method Float printStructureOn:.
3. *This step does not apply to objects for which* isUniqueValue *is true.* Objects that correspond to global Smalltalk names in the original image are mapped to objects with corresponding global Smalltalk names in the target image. This prevents classes and metaclasses from being duplicated. It requires, however, that you be responsible for ensuring that the target image defines all global variables that are referenced (directly or

indirectly) by the object in the source image. If two Smalltalk globals refer to the same object, the result is nondeterministic.

4. Most classes are stored and read in using methods inherited from class Object, which copy instance variables and array elements using instVarAt:, etc. This means that classes must have identical definitions in both the original and target images. It also means that classes that depend on the hash values should be handled specially. Currently, only Set and its subclasses are treated specially for this reason. String and Number (and their subclasses) are treated specially for conciseness of notation (and because SmallInteger must be treated specially anyway). CompiledMethod is also treated specially to ensure the transfer between SOS and LOS images.
5. A receiver's dependents (from the Smalltalk dependency mechanism) are not mapped.

# COPYING CIRCULAR STRUCTURES

The following methods implement a mechanism for copying Smalltalk objects that may contain circularities. The Smalltalk method shallowCopy does not generally copy the complete structure, while deepCopy generally only works for non-circular structures.

## Using the Copying Mechanism

Two *visible* messages are defined to provide the copying of structures.

someObject structureCopy

Answers a copy of the object.

someObject structureCopyWithDict: anIdentityDictionary

Answers a copy of the object, given that a partial list of mappings from objects in the old domain to the new are in anIdentityDictionary. The method may have side effects on anIdentityDictionary, adding new mappings.

The simplest way to use these methods is to use structureCopy. However, if you want to have a handle on the mapping dictionary (either to pre-specify some mappings, to know the mappings after the copy has been created, or to get a copy of several objects that may have common subobjects), you should supply your own IdentityDictionary and use structureCopyWithDict:.

## Implementation Details

This mechanism maps objects based on == (equality). There are a few cautions, however:

1. The copying of objects such as processes will probably cause strange behavior. When a context is copied, the sender field in the new context is nil. The receiver part of a MethodContext, however, becomes mapped to a new object just as any other object

would. CompiledMethods are not copied; rather, the original object is returned. The idea here is that compiled methods should be *constant* objects.

2. Smalltalk treats certain objects in a special way, guaranteeing their uniqueness. These objects in classes such as Boolean, SmallInteger, and Character will return themselves rather than a copy.
3. Most classes are stored and read in using methods inherited from class Object, which copy instance variables and array elements using instVarAt:, etc. This means that classes that depend on the hash values should be handled specially. Currently, only Set and its subclasses are treated specially for this reason.
4. A receiver's dependents (from the Smalltalk dependency mechanism) are not mapped.

# SOS SYSTEM WORKSPACE MODIFICATIONS

The SOS System Workspace has additional text that describes some of the added functionality of this SOS Smalltalk system. The list of globals now includes FontManager, an instance of StrikeFontManager, which maps names to instances of StrikeFont, and, StyleManager, an instance of TextStyleManager, which maps names to instances of TextStyle. The list also includes OSEnvironmentVariables, a Dictionary containing the environment variables passed to the Smalltalk interpreter.

A new section in the System Workspace is titled *Fonts and Text Styles*. This section includes:

```
StyleManager inspect
```

Opens an inspector on all the text styles in the image.

```
StyleManager
    styleName: 'Pellucida Sans-Serif 12 and 14'
    baseNames: #('PellucidaSans-Serif12' 'PellucidaSans-Serif14')
    lead: 3.
```

Installs a new text style containing two fonts. This text style is named 'Pellucida Sans-Serif 12 and 14'. If the fonts are not contained in the image, they will be loaded from the */fonts* directory. The vertical spacing (leading) for this text style is 3 pixels. The text style contains basal, bold, italic, and bold italic faces for each font.

```
StyleManager changeDefaultTextStyle
```

Pops up a menu of available TextStyles. Selecting one progagates it to: default TextStyle, system menus, and all scheduled views and their subviews.

```
Compiler evaluate: ((Disk file: '/smalltalk/textStyles/
pruneToPellucidaDefault08and10.ws') contentsOfEntireFile).
```

Creates a new TextStyle in the Xerox style with mixed serif and sans serif fonts. Discard all other TextStyles and StrikeFonts.

These are additions to the *Display* section:

```
Display setMouseBounds: (-50@-50 corner: 1500@1500)
```

Allows the mouse cursor to move outside the visible screen bounds.

```
TekSystemCall execSystemUtility: '/bin/free' withArgs: (OrderedCollection with: '/dev/disk')
```

Asks the operating system how much space is available on the hard disk.

# MISCELLANEOUS CHANGES

Here is a list of some visible changes to the image not mentioned in any other section:

- The global variable Environment has been renamed OSEnvironmentVariables.
- The use of the writeCloneWithout: message to a system tracer produces a clone but does not produce new source files. New source files may be produced in a separate step.

# CONVERSION OF IMAGES

## Introduction

The conversion software consists of three files, called the Smalltalk Delta files. These are:

- */smalltalk/conversion/deltaT2.1.2ToT2.1.2a.st*
- */smalltalk/conversion/deltaT2.1.2aToT2.1.3.st*
- */smalltalk/conversion/deltaT2.1.3ToT2.1.3b.st*

As you can deduce from the file name, the process of incorporating the file *deltaT2.1.2ToT2.1.2a.st* into a Version T2.1.2 image converts it into a Version T2.1.2a image. The other two files work analogously, of course. To ensure success in converting your image, read this document carefully *before* incorporating any Smalltalk Delta files into your images.

### Should You Convert?

Not everyone needs to follow the conversion procedures in these notes. These notes and the Smalltalk Delta files are provided for Version T2.1.2, T2.1.2a, or T2.1.3 Smalltalk users. (Note that your version number is at the top of your system workspace).

*NOTE*

*If you have not created any Version T2.1.2 or T2.1.2a Smalltalk images that you want to save, it is not necessary to convert your images. Instead, just use the new standard image that is part of the standard software.*

Your standard image should be Version T2.1.3b, the most recent version of the SOS Smalltalk image.

## Saving Disk Space

If you want to free up some disk space, you should consider deleting sources and changes files associated with earlier versions of Smalltalk. For example, if you are not going to keep any T2.1.2 Smalltalk images, you, as system administrator, may also want to recover disk space by deleting the files:

- */smalltalk/system/standardSources.VersionT2.1.2*
- */smalltalk/system/standardChanges.VersionT2.1.2*
- */smalltalk/system/standardSources.VersionT2.1.2a*
- */smalltalk/system/standardChanges.VersionT2.1.2a*

However, if you decide to retain your old work there are two recommended ways to proceed.

- If your work in, for example, Version T2.1.2 is contained in a few new classes of your own creation, or your work is not very large, you may want to fileOut changes from your old image. Filing out changes is a part of standard programming methodology in Smalltalk. Once your changes have been separated from the image, they can easily be incorporated into another image. Add this filed out code to your own copy of the current standard image (Version T2.1.3b).
- Incorporate this update into your old Version T2.1.2 or T2.1.2a image. To do this, proceed to the next heading.

# The FileIn Process

The Smalltalk Delta files, located in the */smalltalk/conversion*, directory are:

- */smalltalk/conversion/deltaT2.1.2ToT2.1.2a.st*
- */smalltalk/conversion/deltaT2.1.2aToT2.1.3.st*
- */smalltalk/conversion/deltaT2.1.3ToT2.1.3b.st*

The order of expressions in the files is determined by dependencies and the Smalltalk class hierarchy.

Smalltalk has several tools to assist you in filing in these update files. A FileList lets you examine the text in a file, make changes, and/or add the code to your image. A ChangeListView partitions code from a file into individual units for fileIn. In this way, parts of a file may be added to your image. A ChangeListView also has a utility for comparing its contents with the

current project's ChangeSet. For a more complete discussion, see the section on the *Change-Management Browser* in Goldberg's *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley, 1984, starting on page 468.

Suppose, after looking at the Class List section, you suspect that a conflict may exist between your additions to an SOS image and the code in the Smalltalk Delta file *deltaT2.1.3ToT2.1.3b.st*

a. Open a ChangeListView and use the middle button to fileIn *deltaT2.1.3ToT2.1.3b.st*. This operation places the code in the ChangeListView but does not incorporate it into the image.

b. Use the middle button to check with system. This operation compares the current ChangeSet to the contents of the ChangeListView. Conflicts are written on an external file.

c. Examine these conflicts using a FileList.

Suppose a conflict exists between your definition of Path scaleBy: and the definition of Path scaleBy: contained in *deltaT2.1.3ToT2.1.3b.st*. You may at this time use a browser to examine users of scaleBy: and decide to discard one of the two methods. Suppose you decide to discard the scaleBy: method in the *deltaT2.1.3ToT2.1.3b.st* file.

a. In the ChangeListView, select the Path scaleBy: method and use the middle button to remove it.

b. Use the middle button menu to do it to each remaining piece, or do all to incorporate all the remaining pieces of code into your image.

If you decide to keep the definition of scaleBy: in the *deltaT2.1.3ToT2.1.3b.st* file and discard your current definition of scaleBy:, simply incorportate all of *deltaT2.1.3ToT2.1.3b.st* by using the do all menu item. This overwrites your version of scaleBy:.

Suppose after finding a conflict you wish to retain the functionality of both versions of the method. If possible, the two methods could be combined into a new method with the same selector. In this case, use the browser to modify your version of the method to have a combined functionality, and use the ChangeListView facilities to prevent the incorporation of the version from the delta file.

Alternatively, it is sometimes difficult to combine conflicting methods. In this case we suggest the following:

a. Use the browser to examine senders of the conflicting method.

b. Create a new method with a different message selector that has the functionality of your version of the conflicting method.

c. Modify appropriate senders to use the new method.

d. Incorporate the code from the delta file into your image, which will overwrite your original version of the conflicting method. Your new method with a different selector will not be affected by incorporating the delta file version of the original method. Your original version of the conflicting method will be overwritten.

# Smalltalk Delta Files Contents

This is a description of the differences between Smalltalk versions. There are only bug fixes in this release. Changes to many classes have been made to fix errors in code and comments.

## SOS Delta File Classes

Here are the classes that have been modified in going from Tektronix Smalltalk Version T2.1.2 to Version T2.1.3.

Behavior
BitEditor class
Browser
ChangeScanner
Character class
Checker class
Class
ClassDescription
Collection
CompiledMethod
Compiler
ContextPart
ControlManager
Cursor class
Debugger
DisplayBitmap class
DisplayScreen
DisplayText class
DisplayTextView
Encoder
FileDirectory
FileList
FileStream
FillInTheBlank class
Float
Form class
FormHolderView
InputSensor
Inspector
ListView
MessageNode
MethodContext
MethodDefinitionChange
MethodNode
NotifierView class
NotifierView
Number
PipeStream
Point
PopUpMenu class
PopUpMenu
PositionableStream
ProcessorScheduler class
ProcessorScheduler
Project
ProjectController class
Rectangle class
Rectangle
ReturnNode
Scanner
ScreenController
ScrollController
SequenceableCollection
Set class
StandardSystemController class
StandardSystemController
StandardSystemView
Stream
StrikeFont
StrikeFont class
StrikeFontManager
StrikeFontManager class
StrikeFont
StringHolderView
Subtask class
SwitchView
SystemDictionary
TekSystemCall class
Text class
TextCollector
TextList class
TextList
TextStyle
TextStyle class

Object
Paragraph
ParagraphEditor
ParagraphEditor class
Pen
PipeReadStream
TextStyleManager
TextStyleManager class
TextStyle
TextView
View

## SOS Delta File Classes

Here is a list of all the classes modified by going from Tektronix Smalltalk Version T2.1.3 to Version T2.1.3b.

Arc
Behavior
Circle
ContextPart
Cursor
Curve
Debugger
Delay class
Delay
DisplayScreen
Explainer
ExternalStream
FileDirectory
FileStream
FileStream class
FillInTheBlankController
Form class
FormEditor class
IdentityDictionary
InputState
Integer
Line
LinearFit
ListController
ListView
NotifierView class
Object
OnlyWhenSelectedCodeController
ParagraphEditor
Path class
Path
Pen
Pipe
PipeReadStream
PipeStream class
Pipe
ProcessHandle
ScreenController
ScrollController class
ScrollController
Spline
StandardSystemView
StrikeFont
StringHolderView
Subtask
Subtask class
SwitchView
SystemDictionary
SystemOrganizer
SystemTracer
TekSystemCall
TekSystemCall class
TextStyle
TextStyleManager class
TextStyleManager
TextView
Time class
WordArray

# System Workspace

Your System Workspace should be updated. If you fileIn all of the code provided in the Delta files, please change the Version in the System Workspace in SOS images to T2.1.3b.

# SMALLTALK DIRECTORIES

With this release of Smalltalk, some new directories and additions and changes to existing directory files have been made.

## New Directories

The directory */smalltalk/conversion* has been added. This contains files for converting one version of a Smalltalk image file into another.

A special directory, */smalltalk/demo/forms*, has been created for forms alone.

New text styles have been added to this release of Smalltalk. Thus, the directory, */smalltalk/textStyles*, has been created and contains code to create instances of text styles in an image. Specifically,

| | |
|---|---|
| PellucidaDefault08and10.ws | Contains code to install the small default style in the Xerox manner, that is, basal, bold, and italic (a triplet) in addition to mixed serif and sans serif faces. |
| PellucidaDefault10and12.ws | Contains the medium sized default faces. |
| PellucidaSans-Serif08tight.ws | Contains an example of minimal vertical spacing. |
| PellucidaSans-Serif.ws | This file along with PellucidaSerifs.ws contain code to create all available text styles in the quadruplet format, that is, basal, bold, italic, and bold italic. |
| PellucidaTypewriters.ws | Contains code to create monospaced fonts. |
| example.ws | Contains code to create a single, large text style in the quadruplet format. |
| pruneToPellucidaDefault08and10.ws | Contains code to remove all text styles and create the small default triplet style. |

## New Files

The following files in the directory, */smalltalk/fileIn*, have been added:

- Graphics-Fractals.st
- addTextStyleToSystemMenu.st
- addTextStyleToYellowButton.st
- extendedBrowser.st
- joydiskAccessAndExample.st
- workspaceFileOut.st

# PRINTING SMALLTALK BITMAP FILES

Look in the */samples/printer* directory for a C program, *bprint.c*, that prints Smalltalk forms or bitmaps on a Tektronix 4644 printer. You can either use this program as it stands if you have the 4644 printer or you can modify the program to be compatible with a different printer.

This program, *bprint.c*, prints Smalltalk bitmaps as generated by the screenCopy menu item or from a writeOn: of a specific form. If you modify the program, the default graphic density and screen width pixels per printer line should be determined by the characteristics of your printer. In *bprint.c*, the default graphic density is double. Option "+s" enables single-density mode which gives you a larger image but with possible truncation.

# Appendix A
# Smalltalk-80 Version T2.1.3b Files

The following is a list of all the files associated with the Tektronix Smalltalk-80 system Version T2.1.3b.

Directory */smalltalk*:

standardImage

Directory */smalltalk/conversion*:

deltaT2.1.2ToT2.1.2a.st
deltaT2.1.2aToT2.1.3.st
deltaT2.1.3ToT2.1.3b.st
structSOSPackage.st

Directory */smalltalk/demo*:

Mastermind-Support.st
Othello.script
Othello.st
Pentominos.script
Pentominos.st
README
WaterJugs.st
demoChanges
demoImage
forms
makingADemoImage.st

Directory */smalltalk/demo/forms*:

aim.form
fall1.form
fall2.form
fall3.form
fractal1.form
fractal2.form
head1.form
head2.form
head3.form
head4.form
head5.form
laundry1.form
laundry2.form
laundry3.form

laundry4.form
laundry5.form
man1.form
man2.form
man3.form
man4.form
man5.form
man6.form
man7.form
man8.form
man9.form
nebula.form
pegasus.form
pendulum1.form
pendulum10.form
pendulum11.form
pendulum12.form
pendulum13.form
pendulum2.form
pendulum3.form
pendulum4.form
pendulum5.form
pendulum6.form
pendulum7.form
pendulum8.form
pendulum9.form
sketch.form
tekLogo.form
usa.form
waterfall.form
wingedHorse.form

Directory */smalltalk/fileIn*:

Animation.st
BookIndexBrowser.st
Clock.st
Examples-Subtasking.st
FinancialHistory.st
Formclass-readMacPaintFile:.st
Graphics-Fractals.st
IconPopUpMenu.st
KineticGraphics.st
PointingHand.st
PopUpMenuHelp.st
ProjectBrowser.st
ProtocolBrowser.st
README
Signals-Support.st
Sound-Support.st

WireList-ASimpleMVCExample.st
addTextStyleToSystemMenu.st
addTextStyleToYellowButtonMenu.st backgroundForm.st
blueInspect.st
corePlot.ws
extendedBrowser.st
findClass.st
hardCopyFunctionKey.st
inspectIt.st
joydiskAccessAndExample.st
sampleBook.index
slideMaker.st
symbolRecovery.st
toothpaste.ws
workspaceFileOut.st
zoomTo.st

Directory */smalltalk/system*:

initialization
standardChanges.VersionT2.1.3b
standardSources.VersionT2.1.3b

Directory */smalltalk/system/initialization*:

SystemWorkspace.ws
black.form
block.form
borderform.form
curve.form
darkgray.form
erase.form
gray.form
in.form
installPellucidaDefault10and12TextStyle.st
lightgray.form
line.form
magnify.form
out.form
over.form
repeatcopy.form
reverse.form
select.form
singlecopy.form
specialborderform.form
togglegrids.form
under.form
white.form
xgrid.form

ygrid.form

Directory */smalltalk/textStyles*:

PellucidaDefault08and10.ws
PellucidaDefault10and12.ws
PellucidaSans-Serif08tight.ws
PellucidaSans-Serifs.ws
PellucidaSerifs.ws
PellucidaTypewriters.ws
XeroxDefault10and12.ws
additionalFonts.ws
example.ws
pruneToPellucidaDefault08and10.ws

# Appendix B
# SOS Image Version Changes

The Tektronix 4404 Smalltalk system supports the Small Object Space image (Version T2.1.3b). The following two lists show the differences between Versions T2.1.2a and T2.1.3, and Versions T2.1.3 and T2.1.3b.

In the left column, you will find the classes or methods that are new or changed. In the right column, you will find an indication of the nature of the change.

Each piece of Smalltalk code in the two lists is categorized as follows:

- NEW – Indicates new code.
- MODIFIED – Indicates a change, possibly a functional change.
- DEFINITION – Indicates a class definition or redefinition.
- EXECUTE – Indicates literal execution of the code.
- REMOVE – Indicates removal of a method.

## Changes Up To Version T2.1.3

The following list shows the changes between the Tektronix Smalltalk-80 images Version T2.1.2a and Version T2.1.3.

| | |
|---|---|
| Behavior kindOfSubclass | REMOVE |
| Behavior removeSelectorSimply: | MODIFIED |
| BitEditor class initialize | MODIFIED |
| Browser removeClass | MODIFIED |
| Browser renameClass | MODIFIED |
| ChangeScanner scanClassExpression:do: | MODIFIED |
| Character class readDefinitionFrom:map: | NEW |
| Checker class classVariablesNotReferenced | NEW |
| Checker class printClassVariablesNotReferencedOn: | NEW |
| Class nonVariableSubclass:instanceVariableNames:c...: | NEW |
| Class replaceNameWith: | NEW |
| ClassDescription definition | MODIFIED |
| ClassDescription kindOfSubclass | MODIFIED |
| ClassDescription moveChangesToSources: | NEW |
| Collection growSize | MODIFIED |
| Collection maxSize | MODIFIED |
| CompiledMethod class newBytes:flags:nTemps:nArgs:nSt... | NEW |
| CompiledMethod class newBytes:flags:nTemps:nStack:nLits: | NEW |
| CompiledMethod class nullSourceDescriptor | NEW |
| CompiledMethod class quickReturnPC | NEW |
| CompiledMethod nullSourceDescriptor | NEW |
| CompiledMethod openByteCodeStream | NEW |
| CompiledMethod setPrimitive: | NEW |

| | |
|---|---|
| CompiledMethod sourceDescriptor: | NEW |
| CompiledMethod sourceDescriptor | NEW |
| CompiledMethod sourceOffset | NEW |
| CompiledMethod trailerSize | NEW |
| Compiler evaluate:in:to:notifying:ifFail: | MODIFIED |
| ContextPart at:put: | MODIFIED |
| ContextPart class basicNew: | NEW |
| ContextPart class new: | NEW |
| ContextPart copy | NEW |
| ControlManager discardCachedDisplayForms | NEW |
| ControlManager restore | MODIFIED |
| Cursor centerCursorInViewport | MODIFIED |
| Cursor class currentCursor: | NEW |
| Debugger bindingOf:forStore: | MODIFIED |
| DisplayBitmap class basicNew: | NEW |
| DisplayBitmap class new: | NEW |
| DisplayScreen class currentDisplay: | MODIFIED |
| DisplayScreen class displayExtent: | MODIFIED |
| DisplayScreen resetFrom:extent: | MODIFIED |
| DisplayScreen resetFrom:extent:offset: | MODIFIED |
| DisplayScreen setDisplayStateFrom: | NEW |
| DisplayScreen setMouseBoundsUpper:low... | MODIFIED |
| DisplayScreen setMouseBounds: | MODIFIED |
| DisplayScreen viewportCenter | NEW |
| DisplayScreen viewport | NEW |
| DisplayScreen writeBitmapOn: | REMOVE |
| DisplayText class text: | MODIFIED |
| DisplayText textStyle | NEW |
| DisplayTextView initialize | MODIFIED |
| DisplayTextView textStyle: | NEW |
| DisplayTextView textStyle | NEW |
| DisplayTextView | DEFINITION |
| Encoder noteSourceRange:forNode: | MODIFIED |
| Encoder sourceMap: | MODIFIED |
| Encoder sourceMap | NEW |
| Encoder tempNames | MODIFIED |
| FileDirectory fullName | MODIFIED |
| FileList createDirectory | MODIFIED |
| FileList createFile | MODIFIED |
| FileList directoryMenu | MODIFIED |
| FileList fileListMenu | MODIFIED |
| FileList fileName: | NEW |
| FileList newFileMenu | MODIFIED |
| FileList resetFileMenu | MODIFIED |
| FileStream appendFileStream: | NEW |
| FileStream binary | MODIFIED |
| FileStream class initialize | MODIFIED |

| | |
|---|---|
| FileStream contentsOfEntireFile | MODIFIED |
| FileStream nextPutAll: | MODIFIED |
| FileStream nextPutAll:startingAt: | NEW |
| FileStream nextPutAll:startingAt:to: | NEW |
| FileStream padTo: | MODIFIED |
| FileStream size | MODIFIED |
| FileStream text | MODIFIED |
| FillInTheBlank class example1 | MODIFIED |
| FillInTheBlank class example2 | MODIFIED |
| Float arcCos | MODIFIED |
| Float arcSin | MODIFIED |
| Float arcTan | MODIFIED |
| Float class initialize | MODIFIED |
| Float class negativeInfinity | NEW |
| Float class notANumber: | NEW |
| Float class notANumber | NEW |
| Float class positiveInfinity | NEW |
| Float exp | MODIFIED |
| Float floorLog: | MODIFIED |
| Float isInfinity | NEW |
| Float isNAN | NEW |
| Float isNegativeInfinity | NEW |
| Float isNormal | NEW |
| Float isPositiveInfinity | NEW |
| Float ln | MODIFIED |
| Float log | MODIFIED |
| Float printOn: | MODIFIED |
| Float printStructureOn: | NEW |
| Float truncated | MODIFIED |
| Float | DEFINITION |
| Form class readFormFile: | MODIFIED |
| FormHolderView cancel | MODIFIED |
| InputSensor currentCursor: | MODIFIED |
| Inspector fieldList | MODIFIED |
| ListView initialize | MODIFIED |
| ListView list: | MODIFIED |
| ListView selectionBox | MODIFIED |
| ListView textStyle: | NEW |
| ListView textStyle | NEW |
| ListView | DEFINITION |
| MessageNode emitForEffect:on: | MODIFIED |
| MessageNode emitForValue:on: | MODIFIED |
| MethodContext adjustPCsForStructReading | NEW |
| MethodContext adjustPCsForStructWriting | NEW |
| MethodContext at:put: | MODIFIED |
| MethodContext basicAt:put: | MODIFIED |
| MethodContext setSender:receiver:method:arguments: | MODIFIED |

| | |
|---|---|
| MethodDefinitionChange accept:notifying: | MODIFIED |
| MethodDefinitionChange sourceFileAndPosition: | MODIFIED |
| MethodNode generateNoQuick | MODIFIED |
| MethodNode generate: | MODIFIED |
| MethodNode sourceMap | MODIFIED |
| NotifierView class openContext:label:contents: | MODIFIED |
| NotifierView class openInterrupt:onProcess: | MODIFIED |
| NotifierView textStyle: | NEW |
| Number class readFrom: | MODIFIED |
| Object shallowCopy | MODIFIED |
| Paragraph recomposeWithTextStyle: | NEW |
| ParagraphEditor changeEmphasis: | MODIFIED |
| ParagraphEditor class initialize | MODIFIED |
| ParagraphEditor emphasisDefault:keyedTo: | MODIFIED |
| ParagraphEditor readKeyboard | MODIFIED |
| Pen mandala:diameter: | MODIFIED |
| PipeReadStream contentsOfEntireFile | MODIFIED |
| PipeStream binary | REMOVE |
| PipeStream contentsOfEntireFile | REMOVE |
| PipeStream text | REMOVE |
| Point negated | NEW |
| PopUpMenu class labels:lines:alignment: | MODIFIED |
| PopUpMenu class labels:lines: | MODIFIED |
| PopUpMenu labels:font:lines: | REMOVE |
| PopUpMenu labels:textStyle:lines: | NEW |
| PopUpMenu markerTop: | MODIFIED |
| PopUpMenu rescan | MODIFIED |
| PopUpMenu reset | MODIFIED |
| PopUpMenu | DEFINITION |
| PositionableStream through: | MODIFIED |
| PositionableStream upTo: | MODIFIED |
| ProcessorScheduler absolutelyTheHighestPriority | NEW |
| ProcessorScheduler class initialize | MODIFIED |
| ProcessorScheduler executeWithoutPreemption: | NEW |
| ProcessorScheduler highestPriority: | MODIFIED |
| ProcessorScheduler resetPriorities | MODIFIED |
| ProcessorScheduler | DEFINITION |
| Project enter | MODIFIED |
| ProjectController class initialize | NEW |
| Rectangle class fromUser: | MODIFIED |
| Rectangle negated | NEW |
| ReturnNode emitForReturn:on: | MODIFIED |
| ReturnNode emitForValue:on: | MODIFIED |
| ReturnNode pc | MODIFIED |
| Scanner scanFieldNames: | MODIFIED |
| ScreenController forkOSshell | MODIFIED |
| ScrollController canScrollDown | NEW |

| | |
|---|---|
| ScrollController canScrollUp | NEW |
| ScrollController canScroll | MODIFIED |
| ScrollController controlInitialize | MODIFIED |
| ScrollController moveMarker: | MODIFIED |
| ScrollController moveMarker | MODIFIED |
| ScrollController scrollDown | MODIFIED |
| ScrollController scrollUp | MODIFIED |
| SequenceableCollection hash | MODIFIED |
| Set class maxSize | NEW |
| Set class new | MODIFIED |
| StandardSystemController class initialize | MODIFIED |
| StandardSystemController textStyle: | NEW |
| StandardSystemController textStyle | NEW |
| StandardSystemView displayBorder | REMOVE |
| StandardSystemView getFrame | MODIFIED |
| StandardSystemView initialize | MODIFIED |
| StandardSystemView label: | MODIFIED |
| StandardSystemView label:style: | NEW |
| StandardSystemView resetLabel: | MODIFIED |
| StandardSystemView resetLabel:style: | NEW |
| StandardSystemView textStyle: | NEW |
| StandardSystemView textStyle | NEW |
| StandardSystemView | DEFINITION |
| Stream do: | MODIFIED |
| Stream nextPutAll:startingAt:to: | NEW |
| StrikeFont asTextStyle | NEW |
| StrikeFont ascentForStdAsciiChars | NEW |
| StrikeFont ascent: | REMOVE |
| StrikeFont bottomLead: | NEW |
| StrikeFont class initialize | NEW |
| StrikeFont class readAll: | NEW |
| StrikeFont class readFrom: | NEW |
| StrikeFont computeAscentDescentForStdAsciiChars | NEW |
| StrikeFont descentForStdAsciiChars | NEW |
| StrikeFont familySizeFace | MODIFIED |
| StrikeFont glyphsSwitchCharacters | NEW |
| StrikeFont initializeFrom: | NEW |
| StrikeFont isFixedPitch | NEW |
| StrikeFont leadInfo | NEW |
| StrikeFont tightLeadInfo | NEW |
| StrikeFont topLead | NEW |
| StrikeFont type: | NEW |
| StrikeFont type | NEW |
| StrikeFont underLineInfo: | NEW |
| StrikeFont writeOnFile: | NEW |
| StrikeFont writeOn: | NEW |
| StrikeFont xTableSwitchCharacters | NEW |

| | |
|---|---|
| StrikeFontManager at:ifAbsent: | NEW |
| StrikeFontManager at:put: | NEW |
| StrikeFontManager checkName: | NEW |
| StrikeFontManager class initialize | NEW |
| StrikeFontManager copy:name:emphasis: | NEW |
| StrikeFontManager errorFontMissing: | NEW |
| StrikeFontManager errorNameFormat: | NEW |
| StrikeFontManager fontNames: | NEW |
| StrikeFontManager install: | NEW |
| StrikeFontManager install:ifAbsent: | NEW |
| StrikeFontManager virtuallyAt: | NEW |
| StrikeFontManager | DEFINITION |
| StrikeFont | DEFINITION |
| StringHolderView displayView: | MODIFIED |
| StringHolderView editString: | MODIFIED |
| StringHolderView initialize | MODIFIED |
| StringHolderView textStyle: | NEW |
| StringHolderView textStyle | NEW |
| StringHolderView | DEFINITION |
| Subtask class copyEnvironment | MODIFIED |
| Subtask class currentEnvironment | MODIFIED |
| Subtask class initializeEnvironment | MODIFIED |
| SwitchView displayView | NEW |
| SwitchView initialize | MODIFIED |
| SwitchView textStyle: | NEW |
| SwitchView textStyle | NEW |
| SwitchView | DEFINITION |
| SystemDictionary appendChangesToSourceFileWithout: | NEW |
| SystemDictionary copyright | MODIFIED |
| SystemDictionary getImageName | MODIFIED |
| SystemDictionary install | MODIFIED |
| SystemDictionary shutdown | MODIFIED |
| SystemDictionary snapshotAs:thenQuit: | MODIFIED |
| SystemDictionary version | MODIFIED |
| TekSystemCall class controlPty:command:mode: | NEW |
| TekSystemCall class createPty | NEW |
| TekSystemCall class execSystemUtility:withArgs: | MODIFIED |
| TekSystemCall class fcntl:function: | NEW |
| TekSystemCall class getMachineType | NEW |
| TekSystemCall class getRealMachineType | NEW |
| TekSystemCall class maxNameSize | MODIFIED |
| TekSystemCall class rump:operation: | NEW |
| TekSystemCall class setMachineType | NEW |
| TekSystemCall class vfork | MODIFIED |
| Text class initTextConstants2 | MODIFIED |
| Text class initTextConstants3 | NEW |
| Text class initTextConstants | MODIFIED |

| | |
|---|---|
| TextCollector defaultContents | MODIFIED |
| TextList class initialize | MODIFIED |
| TextList class onList: | MODIFIED |
| TextList class onList:style: | NEW |
| TextList recomposeWithTextStyle: | NEW |
| TextStyle alignment: | NEW |
| TextStyle alignment | NEW |
| TextStyle asListStyle | NEW |
| TextStyle asMenuStyle | NEW |
| TextStyle basalFontFor: | NEW |
| TextStyle baselineForLists: | NEW |
| TextStyle baselineForLists | NEW |
| TextStyle baselineForMenus: | NEW |
| TextStyle baselineForMenus | NEW |
| TextStyle baseline: | NEW |
| TextStyle baseline | NEW |
| TextStyle boldFontFor: | NEW |
| TextStyle boldItalicFontFor: | NEW |
| TextStyle class default: | NEW |
| TextStyle class default | NEW |
| TextStyle clearIndents | MODIFIED |
| TextStyle defaultFont | NEW |
| TextStyle descent | MODIFIED |
| TextStyle firstIndent: | NEW |
| TextStyle firstIndent | NEW |
| TextStyle flushFonts | NEW |
| TextStyle flushFonts | REMOVE |
| TextStyle fontArray: | NEW |
| TextStyle fontArray | NEW |
| TextStyle fontAt: | MODIFIED |
| TextStyle fontAt:put: | NEW |
| TextStyle fontFor:emphasis: | NEW |
| TextStyle fontFor:face: | MODIFIED |
| TextStyle fontNamed: | MODIFIED |
| TextStyle isFontBoldItalic: | MODIFIED |
| TextStyle isFontBold: | MODIFIED |
| TextStyle isFontItalic: | MODIFIED |
| TextStyle isFontSubscripted: | MODIFIED |
| TextStyle isFontSuperscripted: | MODIFIED |
| TextStyle isFontUnderlined: | MODIFIED |
| TextStyle italicFontFor: | NEW |
| TextStyle leftMarginTabAt: | MODIFIED |
| TextStyle lineGridForLists: | NEW |
| TextStyle lineGridForLists | NEW |
| TextStyle lineGridForMenus: | NEW |
| TextStyle lineGridForMenus | NEW |
| TextStyle lineGrid: | NEW |

| | |
|---|---|
| TextStyle lineGrid | NEW |
| TextStyle listStyleForFont:upperLead:lowerLead: | NEW |
| TextStyle menuStyleForFont:upperLead:lowerLead: | NEW |
| TextStyle nestingDepth | MODIFIED |
| TextStyle newFontArray: | MODIFIED |
| TextStyle nextTabXFrom:leftMargin:rightMargin: | MODIFIED |
| TextStyle outputMedium: | MODIFIED |
| TextStyle outputMedium | NEW |
| TextStyle restIndent: | NEW |
| TextStyle restIndent | NEW |
| TextStyle rightIndent: | NEW |
| TextStyle rightIndent | NEW |
| TextStyle rightMarginTabAt: | MODIFIED |
| TextStyle subscriptedFontFor: | MODIFIED |
| TextStyle superscriptedFontFor: | MODIFIED |
| TextStyle tabWidth | MODIFIED |
| TextStyle unSubscriptedFontFor: | MODIFIED |
| TextStyle unSuperscriptedFontFor: | MODIFIED |
| TextStyle unUnderlinedFontFor: | MODIFIED |
| TextStyle underlinedFontFor: | MODIFIED |
| TextStyle upperLead:lowerLead: | NEW |
| TextStyleManager at:put: | NEW |
| TextStyleManager changeDefaultTextStyle: | NEW |
| TextStyleManager changeDefaultTextStyle | NEW |
| TextStyleManager class flushMenus | NEW |
| TextStyleManager class initialize | NEW |
| TextStyleManager class new: | NEW |
| TextStyleManager fontNamesFromBaseNames: | NEW |
| TextStyleManager fromUser: | NEW |
| TextStyleManager fromUser | NEW |
| TextStyleManager initializeMenus | NEW |
| TextStyleManager removeAssociation:ifAbsent: | NEW |
| TextStyleManager removeKey:ifAbsent: | NEW |
| TextStyleManager styleName:baseNames: | NEW |
| TextStyleManager styleName:baseNames:lead: | NEW |
| TextStyleManager styleName:baseNames:upperLead:lowerLead: | NEW |
| TextStyleManager styleName:fontNames: | NEW |
| TextStyleManager styleName:fontNames:lead: | NEW |
| TextStyleManager styleName:fontNames:upperLead:lowerLead: | NEW |
| TextStyleManager | DEFINITION |
| TextStyle | DEFINITION |
| TextView initialize | MODIFIED |
| TextView textStyle: | NEW |
| TextView textStyle | NEW |
| TextView | DEFINITION |
| View computeInsetDisplayBox | MODIFIED |
| View textStyle: | NEW |

# Changes Between Versions T2.1.3 and T2.1.3b

Here is a list of all classes and methods which are either new or changed from the release of Version T2.1.3 Smalltalk through Version T2.1.3b. The order of methods listed here corresponds to the the contents of the Smalltalk Delta files. The order of the Delta file contents is determined by position in the hierarchy and interdependencies within the code.

| | |
|---|---|
| Arc displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| Behavior whichSelectorsReferTo:special:byte: | CHANGE |
| Circle displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| ContextPart completeCallee: | MODIFIED |
| Cursor centerCursorInViewport | MODIFIED |
| Curve displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| Debugger step | MODIFIED |
| Delay class postSnapshot | MODIFIED |
| Delay class preSnapshot | MODIFIED |
| Delay postSnapshot | MODIFIED |
| Delay preSnapshot | MODIFIED |
| DisplayScreen disableScreenSaver | MODIFIED |
| DisplayScreen getDisplayReport | MODIFIED |
| DisplayScreen setDisplayStateFrom: | MODIFIED |
| DisplayScreen setMouseBounds: | MODIFIED |
| DisplayScreen setMouseBoundsUpper:lowerCorner: | MODIFIED |
| DisplayScreen setViewportLocation: | MODIFIED |
| DisplayScreen viewport | MODIFIED |
| Explainer explainCtxt: | MODIFIED |
| ExternalStream bulkRead:into: | NEW |
| ExternalStream nextBytes:into: | NEW |
| ExternalStream nextNumber: | MODIFIED |
| ExternalStream nextNumber:put: | MODIFIED |
| ExternalStream nextSignedInteger | MODIFIED |
| ExternalStream nextWords:into: | NEW |
| FileDirectory completePathname | MODIFIED |
| FileStream initialize | EXECUTE |
| FileStream appendFileStream: | MODIFIED |
| FileStream asFileDirectory | NEW |
| FileStream backupName | MODIFIED |
| FileStream bulkRead:into: | NEW |
| FileStream class initialize | MODIFIED |
| FileStream freeFileDescriptorFor: | MODIFIED |
| FileStream next:into: | MODIFIED |
| FileStream setName:directory: | MODIFIED |
| FileStream | DEFINITION |
| FillInTheBlankController moveMarker: | NEW |
| Form class readFormFile: | MODIFIED |
| FormEditor class createFullScreenForm | MODIFIED |
| IdentityDictionary keys | MODIFIED |

| | |
|---|---|
| InputState primInputWord | MODIFIED |
| Integer fillBySignExtendFrom: | NEW |
| Line displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| LinearFit displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| ListController viewDelta | MODIFIED |
| ListView deEmphasizeView:andClip: | MODIFIED |
| ListView deEmphasizeView | MODIFIED |
| ListView emphasizeView | REMOVE |
| NotifierView class openContext:label:contents: | MODIFIED |
| NotifierView class openInterrupt:onProcess: | MODIFIED |
| Object exitToDebugger | NEW |
| OnlyWhenSelectedCodeController isControlWanted | MODIFIED |
| ParagraphEditor updateMarker | MODIFIED |
| Path class example | MODIFIED |
| Path displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| Path scaleBy: | MODIFIED |
| Path translateBy: | MODIFIED |
| Pen mandala:diameter: | MODIFIED |
| Pipe fileDescriptor: | NEW |
| Pipe fileDescriptor | NEW |
| PipeReadStream binary | MODIFIED |
| PipeReadStream text | MODIFIED |
| PipeStream class openOnFdn: | MODIFIED |
| Pipe | DEFINITION |
| ProcessHandle resumeProcess | MODIFIED |
| ScreenController forkOSshell | MODIFIED |
| ScrollController class initialize | NEW |
| ScrollController comment | EXECUTE |
| ScrollController controlInitialize | MODIFIED |
| ScrollController initialize | EXECUTE |
| ScrollController scrollDelayLength | NEW |
| ScrollController scrollDown | MODIFIED |
| ScrollController scrollUp | MODIFIED |
| ScrollController | DEFINITION |
| Spline displayOn:transformation:clippingBox:rule:mask: | MODIFIED |
| StandardSystemView validDisplayForm | MODIFIED |
| StrikeFont initializeFrom: | MODIFIED |
| StringHolderView displayView: | MODIFIED |
| Subtask abnormalTermination | MODIFIED |
| Subtask absoluteWait | MODIFIED |
| Subtask class fork:then: | MODIFIED |
| Subtask class fork:withArgs: | MODIFIED |
| Subtask class fork:withArgs:standardIn:standardOut:standardError: | MODIFIED |
| Subtask class fork:withArgs:standardIn:standardOutAndError: | NEW |
| Subtask class fork:withArgs:then: | MODIFIED |
| Subtask class fork:withArgs:withEnv:then: | MODIFIED |
| Subtask class initializeBrokenPipeCatch | MODIFIED |

| | |
|---|---|
| Subtask class initializeWaitManagement | MODIFIED |
| Subtask class initialize | MODIFIED |
| Subtask class install | MODIFIED |
| Subtask class kill: | MODIFIED |
| Subtask class markAndSignalAll | MODIFIED |
| Subtask class terminate: | MODIFIED |
| Subtask class terminateAll | MODIFIED |
| Subtask class waitUpdate | MODIFIED |
| Subtask enhancedPriority | MODIFIED |
| Subtask environment: | MODIFIED |
| Subtask environment | MODIFIED |
| Subtask initBlock: | MODIFIED |
| Subtask initBlock | MODIFIED |
| Subtask install | EXECUTE |
| Subtask interrupt: | MODIFIED |
| Subtask isTerminated | MODIFIED |
| Subtask priority: | MODIFIED |
| Subtask priority | MODIFIED |
| Subtask program | MODIFIED |
| Subtask release | MODIFIED |
| Subtask signalWaitSemaphore | MODIFIED |
| Subtask status: | MODIFIED |
| Subtask status | MODIFIED |
| Subtask taskId: | MODIFIED |
| Subtask taskId | MODIFIED |
| Subtask waitOn | MODIFIED |
| Subtask | DEFINITION |
| SwitchView deEmphasizeView:andClip: | MODIFIED |
| SystemDictionary appendChangesToSourceFileWithout: | MODIFIED |
| SystemDictionary coreLeftLimit: | MODIFIED |
| SystemDictionary lowSpaceNotificationLoop | MODIFIED |
| SystemDictionary postSnapshot | NEW |
| SystemDictionary quit | MODIFIED |
| SystemDictionary saveSpace: | MODIFIED |
| SystemDictionary snapshotAs:onReloadDo: | NEW |
| SystemDictionary snapshotAs:thenQuit: | MODIFIED |
| SystemDictionary snapshotPrimitive: | MODIFIED |
| SystemDictionary snapshot | MODIFIED |
| SystemDictionary versionName | MODIFIED |
| SystemDictionary versionNumber | MODIFIED |
| SystemDictionary version | MODIFIED |
| SystemOrganizer fileOutCategory: | MODIFIED |
| SystemTracer initClampedClasses: | MODIFIED |
| TekSystemCall initialize | EXECUTE |
| TekSystemCall class createDirectory: | MODIFIED |
| TekSystemCall class defaultInterrupt | NEW |
| TekSystemCall class fcntl:function: | MODIFIED |

| | |
|---|---|
| TekSystemCall class freeFileDescriptors | NEW |
| TekSystemCall class ignoreInterrupt | NEW |
| TekSystemCall class initialize | MODIFIED |
| TekSystemCall class pack:intoRegisterWith: | NEW |
| TekSystemCall class read:into: | MODIFIED |
| TekSystemCall class validFileDescriptor: | NEW |
| TextStyle alignment: | MODIFIED |
| TextStyleManager class addMenuDependents: | NEW |
| TextStyleManager class initialize | MODIFIED |
| TextStyleManager class menuDependents | NEW |
| TextStyleManager fromUser | MODIFIED |
| TextStyleManager initializeMenus | MODIFIED |
| TextStyleManager initialize | EXECUTE |
| TextStyleManager | DEFINITION |
| TextView updateRequest | MODIFIED |
| Time class timeWords | MODIFIED |
| WordArray signedIntegerAt: | NEW |
| WordArray signedIntegerAt:put: | NEW |