S C I   C H A N G E S   &   U P D A T E S

in reverse chronological order

The following is a list of changes made to system modules since 2/2/92, the date of the documentation given to Larry Scott by Brian K. Hughes.

8/23/93 Brian K. Hughes

KERNEL.SH

Added the fileCopy subfunction to the FileIO call.

8/16/93 Brian K. Hughes

LOGGER.SC

Added the Reproducible field for SFTracs and eliminated far text.

6/18/93 Brian K. Hughes

MESSAGER.SC

Cleared the oneOnly, killed, and caller properties in the sayFormat method. Failure to do this resulted in the caller not being cleared when a new Cue was made, hence all Cues created by the Messager from the sayFormat method had the same caller as the last one Created by the say method.

6/9/93 Brian K. Hughes

KERNEL.SH

Added the RemapToGray and RemapToPctGray functions to the ColorRemap call.

6/1/93 Brian K. Hughes

LOGGER.SC

Adjusted the output of some of the fields so that the SFTracs program will recognize them. Also added a Reproducible field for SFTracs.

5/13/93 Brian K. Hughes

KERNEL.SH

Added Martin's ColorRemap calls.

3/2/93 Brian K. Hughes

KERNEL.SH

Added subfunction entries for (DoAudio Queue) and (DoSync QueueSync).

2/26/93 Mark Wilden

FILE.SC
KERNEL.SH

A rename method was added to class File to allow renaming of files. A new kernel entry for (FileIO fileRename) was also added.

*2/19/93 Brian K. Hughes

USER.SC

If the event is claimed after going to the iconbar, we return out of User doit: with a TRUE value.

EGO.SC

We no longer look at events in the handleEvent method if the user cannot control. Ego's script will still get events first, however.

CONV.SC

In the MessageObj's showSelf method, the talker (whoSays) is first checked for a value of -1, indicating a "comment" talker. In this case, the caller is cued immediately and no other action is taken.

Also, if the message is not found we no longer exit the game.

INSET.SC

Added explicit return values to the handleEvent method, since the calling object is expecing them (q.v. 1/11/93, GAME.SC).

SAVE.SC

When changing the directory, the size of the input field adjusts to the size of the current directory plus enough space for "\MMMMMMMM". This both A) fixes a bug where the directory name was too long and the extra text trashed memory, and B) allows the user to change the directory down to further levels.

Also, all usages of Print are now clones, which avoids changing Print's default font that the game may have set.

TALKER.SC

The position of the text for viewInPrint talkers was adjusted by the width of the icon to prevent the text from showing over the icon.

SYSTEM.SC

The dispose method of class Script now resets the seconds, cycles, and ticks properties to 0.

1/27/93 Brian K. Hughes

SOUND.SC

The mute method of Sound was using mMUTE as the command to MidiSend, whereas the command is actually mCONTROLLER with a parameter of mMUTE.

1/22/93 Brian K. Hughes

SYSTEM.SH

Added define for module QSOUND back. It was removed because it was thought to be obsolete. Since then a genuine need has arisen for it.

INVENT.SC

The default value for the modNum property of InvItem has been changed from 0 to -1. This makes InvItems more consistent with Feature and makes it possible to use module 0 for the messages. The module will be changed to the current room if the value is -1.

SOUND.SC

The play method used to call the init method unconditionally, which added the sound to the sounds list and inited the C sound object. Due to time issues, the init is now called only if the sound is not already on the sounds list.

1/22/93 Brian K. Hughes [Tester]

TESTMENU.SC

Setting an actor's step size now calls his setStep method instead of just setting the individual properties. This eliminates the bug wherein pressing escape would set the step size to -1, because the setStep method treats a -1 as a "no op".

1/22/93 Brian K. Hughes

LOGGER.SC
SYSTEM.SH
TALKER.SC

Removed the cdAudio global. Instead, the expression (& msgType CD_MSG) is used.

TALKER.SC

The gameTime is reset in the Narrator's init to compensate for times when disk access eats some of the talker's ticks, making it dispose too soon.

[Robert W. Lindsley]
The wait cursor is displayed if the user does not have a mouse
or if the message is a non-modeless CD message. The game's volume is also saved and lowered during CD messages.

KERNEL.SH

Kernel entry #135 was added for SetQuitStr. This call takes a near string as its only parameter, and displays this string on the screen when the interpreter exits.

1/20/93 Brian K. Hughes

INVENT.SC

Claimed the event if the OK button was selected.

1/11/93 Brian K. Hughes

GAME.SC

Messager's sayNext method return 0 in the accumulator if the message was not CD_MSG. This caused Region's doVerb to return 0, which left the event unclaimed, which caused a fall-through default message to be displayed immediately after a room's message. Region's doVerb now returns an explicit value instead of relying on Messager's return value.

1/8/93 Brian K. Hughes

INTRFACE.SC

Code in Dialog's handleEvent method to convert direction events (as would be produced by the joystick) to keyDown/Arrow events had been erroneously removed. Joysticks in dialogs work correctly now.

1/7/93 Robert W. Lindsley

SCALETO.SC

Changed the ScaleTo class to be real-time.

12/30/92 Brian K. Hughes

MESSAGER.SC
TALKER.SC

Messager and Narrator classes are now fully audio-compatable. Messager makes use of the new (Message MsgKey) kernel call, which retrieves the current message pointers from the kernel and stores them into an array, which Messager allocates off heap. The array is then passed to Narrator's startAudio method, via the say method. This gives the audial talkers the information they need for mouth syncs and such, without requiring special calls for audio messages. The message file is used as an index for audio, in that only after the message has been found does the code ask for the current pointers.

Narrator's startAudio method now accepts as its only parameter the address of the message key array (still allocated off heap).

KERNEL.SH

Added an entry for the (Message MsgKey) call.

12/18/92 Brian K. Hughes

USER.SC

If sortedFeatures was FALSE, the event never got sent to addToPics.

EGO.SC

The approachObj bit in ego's state property is now set TRUE in his init. This way, ego will always try to approach objects by default.

12/8/92 Brian K. Hughes

USER.SC

The (User input?) was deleted from the list of conditions by which pMouse would get an event. This means that the cursor may be moved with the arrow keys during handsOff. This was important for being able to use the interactive text.

PMOUSE.SC

The check for the existence of an iconbar and whether the curIcon was the walk icon was removed from PseudoMouse's handleEvent. If the event was a walk, User handles it appropriately before pMouse ever gets a shot. In addition, this bug would not allow the cursor to be moved using the arrow keys if it was the walk cursor (see above change to USER.SC).

FEATURE.SC

Added an setOnMeCheck method that will set the onMeCheck property correctly, according to identifying type. The syntax is:

```
(obj setOnMeCheck: type args)
```

where 'type' is one of ftrDefault (onMe if in nsRect), ftrPolygon (onMe if in polygon), or ftrControl (onMe if on control color). 'args' will be the object ID of a polygon for ftrPolygon type, and multiple control color defines for ftrControl type. ftrDefault type takes no other arguments.

In addition to setting the onMeCheck property, setOnMeCheck also manages the onMeIsControl bit in the state property. This bit eliminates the bug caused when a bit map of controls is the same as the address of an object or $6789 (ftrDefault).

The onMe method was rearranged slightly to better evaluate the three types of checks, and the onMeIsControl bit was checked for ftrControl type.

Also, a new state bit, approachObj, is checked to see if ego should approach objects. If the bit is not set, ego will not approach, regardless of the objects' approachVerbs.

SYSTEM.SH

Added defines for ftrPolygon and ftrControl, and state bit defines for onMeIsControl and approachObj.

12/7/92 Brian K. Hughes

ACTOR.SC

The ignoreBlocks method attempted to delete items from the blocks list without first checking to ensure that the blocks global points to an object. Hence, invoking ignoreBlocks without having invoked observeBlocks would result in a PMachine NAO $0.

GAME.SC

The call to GetCWD in Game's play method has been removed. The interpreter now fills in the curSaveDir buffer with the default directory, which may not necessary be the current working directory.

12/1/92 Brian K. Hughes

INTRFACE.SC

Removed MapKeyToDir in DText handleEvent (not needed). Also, made event global instead of local to check the rectangles.

PRINT.SC

Clones of Print are now stored in temps so that the &rest won't screw up the stack.

11/10/92 Brian K. Hughes

INTRFACE.SC

Called super's handleEvent from DText handleEvent to insure that the return value gets set just as it would if DText had no handleEvent. Also reset the event type to nullEvt and claimed to FALSE so that the dialog will not go away if the event is within a rectangle.

PRINT.SC

All procedures (Prints, Printf, GetInput) now clone Print instead of using the class.

11/6/92 Brian K. Hughes, Robert W. Lindsley, Mark Wilden

GAME.SC

Class Game's doit: method now disposes of a modelessDialog if it runs out of time. This allows modelessDialogs to time out the same way modal dialogs do.

Class Game's replay: method now sets the wait cursor only if user can't input AND user can't control.

INTRFACE.SC

Added support for interactive text in DText class. This required adding a handleEvent method that checks to see if the event is within any rectangles created in the text. If so, the doit method of a global piece of code (pointed to by the textCode global) is called and passed the rectangle number (0 relative). If there is no code, DText will ignore the rectangles.

SYSTEM.SH

Added the textCode global, which points to a piece of code to handle interactive text response.

11/5/92 Brian K. Hughes

FLAGS.SC

A bug in the way in which the array was being accessed in the setSize method was fixed.

GROOPER.SC

If the client's loop was fixed, the doit: method of GradualLooper was bailing out before it's caller was set, preventing the caller from being cued.

11/2/92 Mark Wilden

FILE.SC

The writeString method now returns TRUE if no error condition exists, and FALSE if an error occured during file (or printer) access.

10/29/92 Brian K. Hughes

INVENT.SC

A bug was fixed whereby the select: method of the Inventory's okButton was getting called twice. This meant that the user had to click twice on the button to get out of inventory.

10/28/92 Brian K. Hughes (from Bob Fischbach)

GAME.SC

Added passing &rest to handsOffCode and handsOnCode.

10/27/92 Robert W. Lindsley

ICONBAR.SC

If the user does not have a mouse, the cursor is changed to the INVIS_CURSOR when displaying a help message. The cursor is returned to normal upon dispose of the message.

No-click help is unaffected.

10/26/92 Brian K. Hughes (from Cynthia Hardin)

ACTOR.SC

A small fix was made to the inRect: method that made the bottom and right of the rectangle inclusive. If the point in question is on the bottom or right edge, it will now return TRUE (in the rectangle).

MESSAGER.SC

The talkerList property was removed, and instead of creating a dynamic Set to hold the talkers' ID's an instance of Set is now used. This allows us to override the Set's dispose method to pass along Messager's dispose-WhenDone property.

If the Messager's disposeWhenDone is TRUE, the talkers will be "put to bed" after a sequence of messages is complete. If FALSE, they will stay on the screen. This can be useful when a sequence must be performed manually via a script, but you do not want the talkers to go away (flash) between messages.

10/26/92 Brian K. Hughes

TUTORIAL.SC
ICONBAR.SC
INVENT.SC
SYSTEM.SH

Added Tutorial class. Tutorial is a kind of Script that is meant to be put on the game. It allows the programmer to specify icon items that the user is required to select for the script to continue. There are three methods that the programmer will use:

```
changeState:   used as in a normal script
waitFor:       used to specify which icon item is the 'trigger'
report:        used to display messages in the 'wrong' case
```

As long as the user does the actions specified by the Tutorial, the Tutorial will cue right along, just as a normal script would. If the user makes a wrong choice, the Tutorial can be made to display negative reinforcement messages. The main advantage of the Tutorial class is that it uses the existing IconBar and Inventory.

Here is a sample Tutorial:

```
(instance rm100Help of Tutorial
   (method (report objOrVerb)
      (++ numTries)     <- a property!
      (switchto state
         (
            (switch numTries
               (1    (messager say: HELP NULL NOT_WALK 1)
               (2    (messager say: HELP NULL NOT_WALK 2)
               (else (messager say: HELP NULL NOT_WALK 3)
            )
         )
         (
            ; We will get handed either the incorrect object or
            ;  the incorrect verb
```

```
               (if (IsObject objOrVerb)
                  (switch numTries
                     (1    (messager say: HELP NULL NOT_EGO 1)
                     (2    (messager say: HELP NULL NOT_EGO 2)
                     (else (messager say: HELP NULL NOT_EGO 3)
                  )
               else
                  (switch numTries
                     (1    (messager say: HELP NULL NOT_DO 1)
                     (2    (messager say: HELP NULL NOT_DO 2)
                     (else (messager say: HELP NULL NOT_DO 3)
                  )
               )
            )
         )
      )
      (method (changeState newState &tmp [str 50])
         (switchto (= state newState)
            (
               (self waitFor:
                  (theIconBar at: walkIcon) NULL {Click on the walk icon}
               )
            )
            (
               (Message MsgGet HELP NULL PROMPT 1 @str)
               (self waitFor: ego DO @str)
            )
         )
      )
   )
```

10/23/92 Brian K. Hughes

INTRFACE.SC

Code was added both in Dialog and DItem to make the highlighting feature follow the mouse. This prevents the problem caused when a user moves the highlight to a control using the keys, them moves the mouse over a second control and presses ENTER. Under the old system, the first control would be selected, but now the highlight will follow the mouse to the second control and it will be selected by pressing ENTER.

RANGEOSC.SC

RangeOsc class. Syntax is:

```
   (prop setCycle: RangeOsc howMany firstCel lastCel caller)
```

Additional information is listed at the top of the file.

GAME.SC

Added handsOffCode and handsOnCode properties and handsOff and handsOn methods to class Game. The methods simply alter user control and, in the case of handsOff, kill ego's mover. The methods are setup to check the properties first and, if the property holds an address to an object, call that object's doit: method instead of the default functionality. For example, a specific room might require that some icons be disabled on handsOff. An instance of code in that room could be attached to the

Game's handsOffCode property, and that code will get it's doit: called instead of the default handsOff method's code.

10/20/92 Brian K. Hughes

REGPATH.SC

For whatever reason, this file was not included in the system when it was upgraded in February '92. In addition to the original RegionPath class, the ability to reverse the motion through the points was added by Randy MacNeill.

10/16/92 Brian K. Hughes

FEATURE.SC
WRITEFTR.SC
SYSTEM.SH

A new method, setName, and a new property, state, was added to Feature class. The setName method can be used to change the name of the feature, dynamically allocating space for it off the heap. A new bit, dynamicName, is set in the state property to indicate dynamic name allocation. The dispose method checks this bit to determine if the name needs to be de-allocated.

The Feature Writer takes advantage of this new naming ability to allow users to create features, attach message parameters to them, and have them remain active in the game after the Feature Writer has been disposed.

10/14/92 Brian K. Hughes

ICONBAR.SC

The hide method now checks to ensure that the helpIconItem is an object before trying to turn off its translator bit. This prevents PMachine NAO's in games that do not have help icons in their iconbars.

10/7/92 Brian K. Hughes

GAME.SC

The conditional compile directives were changed to follow the new format (i.e. #ifdef instead of IF).

SYNC.SC

The random loop in MouthSync's doit: method to cover for a sync cel that was invalid has been removed. We can assume that the sync will give MouthSync a valid cel number, so there is no need to check for range violation and generate a random number, which process involved several additional messages and a kernel call.

10/6/92 Brian K. Hughes

KERNEL.SH

Entries were added for two new kernel calls, ShowMovie (#133) and

SetVideoMode (#134).

INVENT.SC

The ESC key now produces the same result as clicking the mouse on the OK button.

9/23/92 Brian K. Hughes

ICONBAR.SC

The iconbar did not transfer the message property of the curIcon (or curInvIcon) to the event properly if the event was a keyDown/ENTER.

GAME.SC

Changed references to the Sounds list (from change of 9/16/92) to refer to the sounds global instead.

ACTOR.SC

In addition to checking the noStepScale bit in the scaleSignal, Actor's doit now also checks to see that the actor is scaling before trying to adjust its stepSize.

TALKER.SC

Added full audio support for talkers.

Also added the ability to control the length of time a talker talks by modifying the value of the textSpeed global. This global can be linked to a slider to let the user control the talker time.

MESSAGER.SC

The oneOnly property now gets set to TRUE in the sayFormat method, which will prevent the messager from calling the sayNext method after the message is done.

FLAGS.SC

The array property is checked in the init method now, and if it is non-zero the flags will not reallocate new space. Similarly, the array property is reset to 0 in the dispose method.

INVENT.SC

The nsTop property of IconItem, which defaults to -1, is now set to 0 in the drawInvWindow method of Inventory, just prior to determining its final position. This prevents the top row of pixels in the IconItem's cel from being overwritten by the window itself.

9/16/92 Brian K. Hughes

GAME.SC

The Sounds list was misspelled several places, leading to confusion between it and the Sound class.

9/10/92 Brian K. Hughes

GAME.SC

Game's doit: method now exits early if a fastCast exists after the call to Animate. This would indicate that a talker has been initiated from someone's doit, and that next cycle the game will lock itself into the fastCast processing loop. Exiting the doit method early prevents the user from getting a doit, which can lead to unexpected results if there is an event waiting in the queue.

MESSAGER.SC

The Messager now passes messager parameters to Talker, which will use them ONLY if the message has an audio component. This allows the talker to pass the parameters along to the DoAudio call. For non-audio talkers, the extra parameters are ignored.

Note: This does not enable you to use the talker independent of Messager.

PRINT.SC

Print's font now defaults to the userFont for each item that uses a font. This prevents having to specify (Print font: userFont) in the game's init method.

Also removed some debug code from addTitle.

SYSTEM.SH

A complete list of resource type defines was added, using a new naming convention. Resource type defines will now be prefixed with 'RES_' to differentiate them from other defines. The original list has been left in for backward compatability, but will be removed shortly.

SYNC.SC

Various small changes, mostly by Dan Carver.
Largest change (and most significant) is that the syncs are now identified by standard message parameters instead of message number.

9/2/92 Brian K. Hughes

ACTOR.SC

Modified all the code that sets the scaleSignal to make sure it ORs the appropriate information into the existing scaleSignal, instead of just assigning the property to the new value. This insures that any other bits set in the scaleSignal (like noStepScale) will be preserved.

GAME.SC

The doit method of game is now exited if there is a fastCast after the call to Animate.

This fixes a very obscure bug, in which a call to Messager say: originated from the doit of a cast member. After the call to Animate, there will be a fastCast (because the talker created it), yet the game's doit: is still not done with the current cycle. At the end of the cycle the doit method of User is called, which polls for a new event. If there is an event waiting, new processes with new messages may be spawned even though there is a fastCast waiting.

This fix is obsolete as soon as the new versions of Talker and Messager (no fastCast!) are checked in.

8/31/92 Brian K. Hughes

ACTOR.SC

Moved the (avoider doit:) code outside the cond in which the (mover doit:) code resides in Actor. The actor's avoider now gets a shot regardless of whether or not the actor has a mover.

A new scaleSignal bit, noStepScale, was added. If this bit is on, the actor will not recalculate its stepSize each doit.

SYSTEM.SH

Added the define for noStepScale and the define for the FLAGS module.

SOUND.SC

The mute method of Sound was changed to allow a single channel to be muted. In addition, the same code (MidiSend) is now used in a loop to mute all channels, allowing us to remove the MuteSound procedure from MIDI.S. The format for the mute method is now:

```
      (sound mute: trueOrFalse [channel])
```

If 'channel' is not specified, the mute/unmute operation will be applied to all channels.

8/25/92 Brian K. Hughes

MESSAGER.SC

In Messager's sayFormat method, &rest was not being passed to the FindFormatLen procedure and Format kernel call, thus only the first argument passed was being formatted into the control string.

8/21/92 Brian K. Hughes

SYSTEM.SC

The EventHandler class was checking for (event claimed?) instead of (evtClone claimed?) as one of the conditions for dropping out of the loop. Also, made sure to dispose of cloned event before returning.

PAVOID.SC

Made sure blocker has a mover property before trying to access it. This bug caused PAvoider to PMachine if the object blocking ego was not an Actor or subclass of Actor.

8/21/92 Brian K. Hughes

FEATURE.SC
EGO.SC

Made the handleEvent method actually claim the event instead of just return a TRUE if the event should be claimed. (Reverse of change made 7/31/92)

8/21/92 Brian K. Hughes

POLYEDIT.SC

References to currentPalette, an obsolete global, have been removed.

8/20/92 Brian K. Hughes

ICONBAR.SC

The iconbar now references the loop used for the current inventory item with no modifiers. The code used to always add 1 to the loop number because it assumed a specific order to loops in the inventory view.

The gameTime global variable is now updated inside the iconbar's doit loop. This ensures that talkers and other real-time objects launched from the iconbar (or subclass) will set their timings correctly.

A bug was fixed wherein if the iconbar's curInvIcon is 0 the iconbar will not try to reference its type or message.

8/19/92 Brian K. Hughes

GAME.SC

Removed subtitleLang and parseLang properties and setSpeed, checkAni, and setFeatures methods from Game. Also removed locales list and RoomControls.

Also, the code throughout the module has been straightened up, made more consistent in form, and comments added or updated.

SYSTEM.SC

Made the EventHandler class clone its event before sending it to its elements. This avoids the problem of one of the elements executing a ((user curEvent?) new:) and wiping out the EventHandler's event.

SIGHT.SC

A reference to egoBlindSpot, an obsolete global, has been removed.

INSET.SC

References to currentPalette, an obsolete global, have been removed.

ACTOR.SC

Removed palette property and isExtra method from View, and findPosn method from Actor. Also removed the call to SetNowSeen in Prop's doit.

SYSTEM.SH

Removed obsolete module defines, including: AVOIDER, SORTCOPY, TEXTRA, MOUSER, QSCRIPT, TIMEDCUE, QSOUND, LASTLINK, PRINTD, PCYCLE, LANGUAGE.

Also removed obsolete global variables, including: speed, showStyle, locales, aniThreshold, sortedFeatures, egoBlindSpot, demoDialogTime, currentPalette, and theMenuBar.

Also set the default for msgType to be TEXT_MSG.

8/18/92 Brian K. Hughes

DOOR.SC

Door now uses module pointed to by modNum property instead of DOOR define.
If modNum property is -1 (default), it is set to curRoomNum.

8/17/92 Brian K. Hughes

DLGEDIT.SC
POLYEDIT.SC
WRITEFTR.SC
TALKER.SC

Replaced setting the title property in Print directly with invocations of the addTitle method. Doing otherwise will cause an error because Print assumes that the title property contains a pointer to allocated heap.

TALKER.SC

The ability to pass message parameters directly to a talker has been removed. The code was built originally to provide some measure of backward compatability, which is no longer necessary. The say: method of Narrator and Talker now takes a single argument, which will be a buffer (for text messages) and probably an audio number for audio messages.

MESSAGER.SC

As above, all code that passed message parameters to a talker was removed.
Also, error messages were made more clear and the findTalker method now returns the narrator by default, after displaying a warning message.

8/14/92 Brian K. Hughes

MESSAGER.SC

The oldIconBarState was not being set in the sayFormat method, as in the say method.

8/14/92 Brian K. Hughes

ICONBAR.SC

Code was added to the hide method to turn off the TRANSLATOR bit in the helpIconItem's signal. This fixes a bug wherein the helpIconItem was still active when entering the iconbar if it was not used before exiting the last time. This only happened if the NOCLICKHELP bit was not set in the iconbar.

8/12/92 Brian K. Hughes

PRINT.SC

A new procedure, FindFormatLen, was added. Given a control string and its parameters, FindFormatLen will return the number of bytes that the formatted string will require (at maximum).

The addTextF method of Print was modified to make use of this procedure.

The addition of this procedure means that it is no longer required to pass a pre-defined buffer to either Printf or addTextF.

8/12/92 Brian K. Hughes

MESSAGER.SC

The sayFormat method was altered to make use of the new FindFormatLen procedure, instead of repeating the same code.

8/12/92 Brian K. Hughes

USER.SC

Code was added for speech support. First, User checks for the existence of a speechHandler, and if so lets it handle the event. Otherwise, the event is processed according to its type and message.

Also, a check for (user canControl) was removed from the cond for direction events; if user can't control, only ego needs to know. This bug made it impossible to move the cursor using the keyboard if the user did not have control, even if it was not a walk cursor.

8/12/92 Brian K. Hughes

ICONBAR.SC

The findIcon method only checked as many elements of the IconBar as the number of parameters you passed. It now checks all elements of the list.

8/12/92 Brian K. Hughes (from Chad Bye)

KERNEL.SH

Obsolete kernel functions were commented out. These include Parse, Said, SetSynonym, ShowObjs, and StrSplit.

New kernel functions were added, including: AssertPalette, TextColors, TextFonts, Record, and PlayBack.

8/12/92 Brian K. Hughes

PAVOID.SC

If there are no obstacles in the current room, then the avoider's client's mover would have no obstacles. This condition was not handled in PAvoider's doit. The code now creates a list for the mover's obstacles if there is none and places the new polygon into it. The first doit cycle wherein the PAvoider's client is > 20 pixels away from the blocker, the list is disposed to prevent frags.

8/12/92 Brian K. Hughes (from Martin Peters)

MOTION.SC

The initial value of cycleCnt for cyclers was changed from (- gameTime cycleSpeed) to gameTime. This forces the cycler to wait for a time equal to the client's cycleSpeed before cycling the first time. This bug caused the first cel of an animation loop to be skipped over very fast.

8/12/92 Brian K. Hughes

GAME.SC

A call to the obsolete kernel call SetSynonyms was removed.

8/11/92 Brian K. Hughes

LOGGER.SC

The 3-character QA initials field was changed to an 8-character login name. This is to facilitate the new bug reporting system from Dynamix.

8/11/92 Brian K. Hughes

FILE.SC

The name property had been redefined and set to 0, which caused instances of the File class to become invisible in the object list. This property definition was removed, allowing the name to default to the class name.

8/10/92 Brian K. Hughes

PRINT.SC

Added an addTitle method, which takes the following forms:

```
(Print addTitle: @buffer)
(Print addTitle: noun verb case sequence module)
```

8/10/92 Brian K. Hughes

DLGEDIT.SC

Window title can now be expressed as message parameters, as well as a literal string. DialogEditor now writes title to SCI file.

8/10/92 Brian K. Hughes

SCALETO.SC

A new class, ScaleTo, has been added to the system. ScaleTo scales its client from its current size to the specified size over a period of time. Usage of the ScaleTo class is:

```
(object setScale: ScaleTo newSize [caller])
```

8/10/92 Brian K. Hughes

MESSAGER.SC
TALKER.SC

Checks were added to ensure that theIconBar points to an object before making references to it.

8/10/92 Brian K. Hughes

STOPWALK.SC

Code in the doit method that killed the client's mover was removed. This code was not necessary for StopWalk to work correctly, and in fact made it impossible to have a StopWalk cycler and Follow/PFollow mover on an actor at the same time.

8/10/92 Brian K. Hughes

MESSAGER.SC

The ability to say a range of sequences has been added to the say method. The parameter format for the say method is now:

```
(messager say: noun [verb [case [sequence [caller [module]]]]])
(messager say: noun [verb [case [fromSequence toSequence [caller [module]]]]])
(messager say: NEXT [caller])
```

A frag caused by not deallocating heap space in sayFormat was fixed.

8/07/92 Brian K. Hughes

POLYEDIT.SC

The PolygonEditor now handles editing polygons on the altPolyList. They are read into the editor as normal polygons, but a new property called srcList will be set to indicate that the polygons come from the altPolyList. The polygons can be distinguished by different colored lines in the control screen, but as yet they are the same color on the visual screen. This is due to the fact that different palettes yield unpredictable colors.

8/07/92 Brian K. Hughes

PCHASE.SC

The init methods of the PChase and PFollow classes was modified to call the super's with variable parameters, depending on the number of parameters passed to the init. This allows the re-initing of the mover done at each node and when step size changes to still use the existing path without adjustements. The most common manifestation of this bug was the tendency for the chasing actor to cut through polygons.

8/03/92 Brian K. Hughes

TALKER.SC

The cueVal property of Narrator was never being reset to 0, causing a talker that had been cancelled with the right mouse (or ESC) to continue to display only the first message in a series. The cueVal is now reset to NULL in the Narrator's dispose.

8/03/92 Brian K. Hughes

GAME.SC
MESSAGER.SC
ICONBAR.SC
SYSTEM.SH

A new class, Cue, was created for the purposes of delayed cuing. Global 18, cuees, may be used as a pointer to a set that contains Cue clones. At the beginning of each Game doit and IconBar doit cycle, the list is checked and a doit: message is sent to each of its members. The doit: of Cue simply cues the cuee, with a register and cuer as arguments.

This system allows Messager to cue its client after the stack has returned. The problem was that Messager could cue a script, which could call Messager again, without allowing the original talker to dispose correctly.

7/31/92 Brian K. Hughes

ACTOR.SC (from Dave Artis)

Changes were made to the order and nature of the step size calculations in Actor's doit: method. These changes help reduce integer overflow problems and keep the actor from spinning or "getting stuck" at very small sizes.

FEATURE.SC
EGO.SC

A temporary variable was added to the handleEvent methods to hold the return value. Simply returning (event claimed?) is no longer reliable, since the events used in the IconBar, GameControls, and Inventory are no longer clones. This new variable ensures a proper return value from the handle-Event method regardless of the current condition of uEvt.

7/28/92 Brian K. Hughes

FEATURE.SC

If no argument is passed to a feature's facingMe: method, the actor to face is assumed to be ego. The method will now return TRUE if this is the case AND ego is not in the cast. This prevents bugs caused by ego trying to face even though he's not in the cast.

7/27/92 Brian K. Hughes

ICONBAR.SC
INVENT.SC
GCONTROL.SC
USER.SC

The new: method of uEvt (in USER.SC) was overridden to simply poll for a new event record, not actually create a clone. The IconBar class and its sub-classes were modified to use ((user curEvent?) new:) instead of (Event new:). This reduces the liklihood of errors caused by the constant creation and destruction of dynamic events. In addition, (user curEvent?) now always points to the current event.

7/24/92 Brian K. Hughes

RANDCYC.SC
TALKER.SC

The old RandCycle class from RANDCYC.SC was replaced with the RTRandCycle class from TALKER.SC. This brings RandCycle up to real-time. In addition, a new property called "reset" has been added to RandCycle which, if true, will set the client's cel to 0 upon init and cycleDone. Talkers use this feature. The parameter list for RandCycle is now:

```
(prop setCycle: RandCycle timeToCycle [caller [resetOrNot]])
```

7/24/92 Brian K. Hughes (from Kevin Ray/Oliver Brelsford)

EGO.SC

Overrode the facingMe method to always return TRUE. This prevents ego from turning around when the player clicks on him in an attempt to face himself.

7/23/92 Brian K. Hughes

USER.SC
EGO.SC
ACTOR.SC

The Ego class was split out of USER.SC into its own module, EGO.SC. In addition, the setSpeed of Ego was moved into the Actor class.

7/22/92 Brian K. Hughes

GAME.SC

In Game's replay method, the style used to redraw the pic is changed to PLAIN if it was originally one of the scrolling styles, or if it contained either HMIRROR or VMIRROR. Since the default style for a room is -1, this change prevents mirrored pictures from drawing unmirrored and scrolling pictures from scrolling in from nowhere.

7/21/92 Brian K. Hughes

ICONBAR.SC

A change that was forgotten in the work done on 7/08/92. The IconBar now sets the cursor to ARROW_CURSOR before calling the doit, not normalCursor.

7/20/92 Brian K. Hughes

MESSAGER.SC
TALKER.SC

A new method, sayFormat, was added to Messager. The sayFormat method takes the following syntax:

```
   (messager sayFormat: talkerDefine @controlString formatParams [caller])
```

It will format the string into a buffer of the appropriate size, then pass the string to the talker. The say: method of Narrator was modified to handle a single argument (a string pointer). Typical uses would be:

```
   (messager sayFormat: GATE_GUARD {How are you, %s?} @egoName self)

   (messager sayFormat: SALESMAN {That will be %d dollars for the %s.}
      (theItem cost?)
      (theItem name?)
   )
```

Note that the talker define is required, since the messager will not be getting the message from a message file. The (Message MsgGet) kernel call will return the talker number.

7/20/92 Brian K. Hughes

MESSAGER.SC

Removed the one-cycle delay before disposing; messager used to put itself on theDoits list, then dispose next cycle. Messager now just disposes directly, when necessary.

Also code added 4/08/92 (q.v.) to ignore messages whose talkers are -1 will be checked in now.

7/10/92 Brian K. Hughes

PATH.SC

In the init method, properties are set only if arguments are passed. This prevents properties from getting set to the next available garbage on the stack when the Path is inited with no arguments (such as from Actor's setStep method).

7/09/92 Brian K. Hughes

INVENT.SC

The advanceCurIcon method was removed, allowing the IconBar's default method to be used. The method was over-ridden to allow the user to select the help icon by cycling through the icons. The removal was done in order to make the IconBar and its subclasses consistent.

POLYPATH.SC
SYSTEM.SH

A new global, altPolyList (95), was defined to hold a pointer to a list of alternate polygons. PolyPath was modified to check this list in addition to the room's obstacles list. The function of PolyPath is now:

1) Generate an optimized path, using the room's obstacles list
2) Get the next target point in the path
3) Generate an unoptimized path to this point, using the altPolyList
4) If the returned end point is not the same as the point from step 2, the original path is blocked by an alt poly; therefore, insert an end-of-path marker ($7777) at this point
5) If not end-of-path, go to step 2

This change allows us to have a second list of polygons without requiring that they be merged into the room's obstacles list. Alt polys are highly useful with Doors (see below).

DOOR.SC

This is a new class. It is based on similar Door classes by Al Lowe and the Iceman group.

By default, Doors create a alt polygon around them to keep actors from walking through them. When the door opens, the alt poly is taken off the altPolyList, allowing free passage. When the door closes, it puts its alt poly back on the altPolyList. This has two major advantages:

1) There is no way an actor can be "chicken-walked" around a door's control color block by being repositioned inside the room's poly
2) Since control color blocks are eliminated, room's polygons can be made much closer to diagonally positioned doors

Door doco is forthcoming.

TALKER.SC

Code was added to hide the cursor (set to INVIS_CURSOR) when a talker comes up, but only if the player has no mouse. This prevents the wait cursor (or similarly-large cursor-like object) from hiding the text.

Also, the time that a talker's mouth moves was lengthened by 33%.

7/08/92 Brian K. Hughes

GAME.SC

The code in the replay method of Game that sets the cursor was expanded to allow for a handsOff condition. This ensures that the wait cursor will be displayed if restoring a game that was saved during handsOff.

Code was added to the save & restore methods of Game that will check for to see if the current save directory is valid, and if not ask for a new path. The code will loop until a valid path is entered.

FEATURE.SC

In the handleEvent method, if we determine we will not approach the feature, we now call facingMe before calling the changeState method of CueObj. This allows ego to face the object before we invoke the doVerb.

The facingMe method now skips the actual facing code (in the notFacing method) if the sightAngle is equal to the feature default ($6789).

GCONTROL.SC

Removed obsolete references to helpStr, replacing them with the equivelant checks using helpVerb. Also, used Print with a set width and font instead of the previous Format/Prints usage.

ICONBAR.SC

Upon exiting the iconbar code, the cursor is now set to the waitCursor if the player does not have input and control.

When checking for a keyDown in the dispatchEvent method, the code was still using the old (icon select:) methodology instead of the new
(if (icon select:) (icon doit:)) methodology.

The Print statement that displays the help message was made better looking.

INTRFACE.SC

Optimized use of event's properties by using temp variables.

LOGGER.SC

Reformatted the Print displays so that they are better spaced.

PRINT.SC

Added an optional fourth parameter to the GetInput procedure that specifies a font other than the default SYSFONT.

SAVE.SC (changes by Gary Kamigawachi)

All text now resides in the message file. Dialogs pull messages for text, button text, titles, etc.

STOPWALK.SC

Oddly enough, when the client begins to walk and is using a single view for both walking and stop loops, there was no code to set his loop correctly for walking. It was left up to the setHeading/setDirection code to handle this, which caused occasional "pirouetting".

USER.SC

Checks were added to ensure that the player has control (user canControl:)

before processing direction events, and that the player has input (user canInput:) before processing keyDown and mouseDown events. These checks had gotten deleted erroneously during the first revision of User.

7/08/92 Brian K. Hughes

KERNEL.SH

The enum PALVARYNEWTIME was added to the PalVary entry.

7/02/92 Brian K. Hughes

ACTOR.SC

Fixed a bug in the match code of Prop setScale. The new (clone) scaler was being improperly created, resulting in various errors.

Also, moved code that set the actors step sizes (Actor doit:) to before the mover's re-init. This ensures that the InitBresen code will have the proper step size values to work with. The mover is now only re-inited if it decends from MoveTo or PolyPath.

6/05/92 Brian K. Hughes

SAVE.SC

Adjusted positions of yes/no buttons in the dialog for confirming the deletion of a save game.

6/04/92 Brian K. Hughes

SAVE.SC

Escape key now produces a -1, not 0. Changed the code in SRDialog to look for the right return value.

6/04/92 Brian K. Hughes

GAME.SC

Added Mark Hood's fix for addToObstaclesCode for addToPics. A logic error caused addToPics to never create polygons around themselves.

Moved buttons in error dialogs down so it doesn't appear over the text.

In class Region method dispose, there is code to check for a timer attached to the region and dispose of it. Since Timer dispose: doesn't remove it from the timers list, the timer's delete: method is now called as well.

5/28/92 Brian K. Hughes

KERNEL.SH

Added new kernel call ResCheck, which will allow the programmer to verify the existence of a resource without loading it.

Added a fifth function to PalVary, PALVARYTARGET, which allows the programmer to insert a custom color into the target palette. More doco is forthcoming in the systems group report.

5/28/92 Brian K. Hughes

USER.SC

Conditioned whether pMouse gets dirStop events by whether the event type is a keyDown. This fixed a bug wherein the joystick never produced dirStop events, making it impossible to stop the cursor with the joystick.

5/28/92 Brian K. Hughes

ICONBAR.SC

In the noClickHelp method, the temporary event was not being disposed if the noClickHelp was terminated by clicking on the help icon again. If it was terminated in any other way, the dispatchEvent method will dispose the event, as before.

5/11/92 Brian K. Hughes

SYSTEM.SC
ICONBAR.SC

isMemberOf now returns TRUE if the object in question is an instance of or the actual class being compared. This fixed the bug in IconBar wherein IconBar was mysteriously not a member of IconBar.

5/11/92 Brian K. Hughes

PRINT.SC
INTRFACE.SC

Prints with no active items will now return FALSE if the ESCAPE key was pressed and TRUE if the ENTER key was pressed.

5/09/92 Brian K. Hughes

SAVE.SC

Made sure the current directory shows up in the edit box when asking for a new save game directory. This got deleted in the conversion to the new Print object.

5/06/92 Brian K. Hughes

ICONBAR.SC

Removed the kludge for testing whether or not to localize the event in the noClickHelp method (q.v. 2/26/92). The isMemberOf method of Object is functioning normally now (see below) so the kludge is not necessary.

5/06/92 Brian K. Hughes

SYSTEM.SC

Changed the isMemberOf method of class Object to also return TRUE if the object tested is equal to the class tested against. For example, (IconBar isMemberOf: IconBar) will return TRUE.

4/30/92 Brian K. Hughes

STOPWALK.SC

Added a (super doit:) in StopWalk doit so that an actor who is stopped and has a separate stopped view will still cycle. If you don't want the actor to cycle, you can reduce each of the stopped loops to one cel. Having an entire stopped loop of cels can be useful in situations such as ego standing holding a moving animal.

4/30/92 Brian K. Hughes

GAME.SC
GCONTROL.SC

Added two properties, panelObj and panelSelector, to class Game. Also added some code in the doit method of Game that checks the panelObj property and, if set, sends a 'panelSelector' message to panelObj. This change allows a major heap-saving feature, which is that the control panel can now be contained in an external module, called in when it is used, and unloaded from memory BEFORE the user's action takes place.

For example, in Laura Bow II the control panel requires about 1300 bytes of heap. If the save button is pressed, the panelObj and panelSelector properties of Game are set to theGame and #save, respectively, and the control panel is unloaded from memory, regaining the 1300 bytes. At the beginning of the next doit, the panelObj and panelSelector properties are evaluated and the save dialog is brought up, which requires about 1600 bytes of heap. The result is that saving the game requires only 1600 bytes maximum instead of 2900 bytes.

The game's panelObj and panelSelector properties are now set in the ControlIcon's doit method.

4/30/92 Brian K. Hughes

PRINT.SC

Fixed bug in addIcon whereby parameters were being used even if no values were passed.

4/30/92 Brian K. Hughes

ACTOR.SC

The step size calculations in Actor's doit method for scaling actors has been readjusted. The old calculation rounded off, while the new calculation rounds to the nearest integer. This creates a much more realistic look for a scaling actor.

4/23/92 Brian K. Hughes

FEATURE.SC
GAME.SC

Changed default modNum in Feature and Region from 0 to -1 to allow use of message file number 0 as an object's default.

4/19/92 Brian K. Hughes

ACTOR.SC

Made sure to turn off the viewAdded bit of an addToPic's signal when it is deleted from the addToPics list. This way, if the view is still in memory when it is used again, it won't automatically assume it is intended to be an addToPic again.

Added a parameter to Actor's setStep method, leaveOriginal. If a TRUE value is passed in this parameter, the origStep property of Actor will NOT be updated to the new xStep and yStep values. So far, this is used only in Actor's doit method for scaling.

4/16/92 Brian K. Hughes

WRITEFTR.SC

Removed references to PicView.

4/16/92 Brian K. Hughes

TALKER.SC
PRINT.SC

Since talkers' Prints are modeless, we can't dispose of the window clone created in the talker's display method immediately after the Print. Instead, the Print's cue method (which gets invoked from its dialog's dispose) gets rid of any windows that are connected to Print.

4/15/92 Brian K. Hughes (from Bob Fischbach/Hugh Diedrichs)

MESSAGER.SC

Removed code that checked for a non-zero fifth parameter to Messager say (the module number), and just checked for a fifth parameter. This allows you to send 0 as a legal module number.

4/14/92 Brian K. Hughes

GCONTROL.SC
ICONBAR.SC

Changed the select method of ControlItem to a doit, which gets called from the dispatchEvent method of IconBar (or subclass thereof). This allows the select to determine actions to be taken, then the actions to actually happen near the end of the dispatchEvent method, allowing the IconBar (or subclass) to be disposed first. For example, the save icon in the control panel will now allow the control panel to be disposed before the save code is brought in, reducing the amount of heap required to nearly 1/2.

4/13/92 Brian K. Hughes

PRINT.SC

Made sure to set the port back after a modal print.

4/11/92 Brian K. Hughes

PRINT.SC

Reset the window property back to 0 when disposing. This prevents the Print class from trying to pass it along to the dialog again next time.

4/11/92 Brian K. Hughes

TALKER.SC

Removed the restriction that colors wouldn't be set for custom windows. The only danger in this is that completely custom windows (those that draw their own background rectangles) won't change color automatically to match the text background.

4/09/92 Brian K. Hughes

FEATURE.SC

Added a method, initialize, which performs the appropriate feature initializer code. The init method now invokes this method before dealing with adding the feature to the cast or features list. This way, a view may have the appropriate feature initializer code done without having to call the init and add it to a list (useful for views that will become addToPics, see below).

ACTOR.SC

Removed PicView class. The view class now assumes all functionality of the PicView class. Views may be added to the pic with the addToPic method (as before), but now the view itself is put onto the addToPics list, not the features list (see below).

To make a view add itself to the pic automatically upon init, set the "viewAdded" bit in the view's signal, or invoke the addToPic method instead of the init.

USER.SC

Added the addToPics list to those considered by OnMeAndLowY during the sorted features process.

GAME.SC

The members of the addToPics list now get deleted on a room change, as well as disposed.

4/08/92 Brian K. Hughes

PRINT.SC

Check the message size to see if the message is actually in the message file, and if not don't try to allocate heap. This prevents a "zero heap allocation request" error.

4/08/92 Brian K. Hughes

MESSAGER.SC

Messager now ignores a message if the return value from the findTalker method is -1. This value can be used to indicate a talker that is used purely for comments in the message file, talkers that perform some other action besides displaying a message, et al.

4/08/92 Brian K. Hughes (from Dave Artis/Oliver Brelsford)

SCALER.SC

Reorganized calculations in init and doit to optimize for speed, while still attempting to retain some of the efficiency. The doit method was modified to make sure the calculations were done even if ego is out of range. Also, the doit is invoked from the init so that the correct parameters are set initially.

4/08/92 Brian K. Hughes

ACTOR.SC

Solidified parameters passed to View and Prop's setScale method and added new functionality. The possible forms of setScale are:

```
View:
   setScale: (no args)     - sets scaleSignal to scalable only
   setScale: 0             - turns off all scaling
   setScale: n             - sets auto-scaling based on vanishingY
                               and ego's size of 100% at y = 'n'

Prop:
   setScale: obj           - sets scaler object (with optional params)
   setScale: MATCH obj     - matches scaling of 'obj'
```

4/06/92 Brian K. Hughes

SYSTEM.SH

Replaced define RFEATURE with GCONTROL (978). RFeature and RPicView classes are obsolete. GameControls has been split out of SLIDICON and put into GCONTROL.

Also, the define NEXT (used with messager) was changed from 0 to -1 so it would not conflict with ALL.

4/06/92 Brian K. Hughes

ICONBAR.SC

In noClickHelp, if the item under the mouse does not have a helpVerb, it will not try to print a help string.

4/05/92 Brian K. Hughes (from Hugh Diedrichs)

ICONBAR.SC

Set the event's message to (| userEvent helpEvent) in IconBar's doit. This allows the normal help to work correctly, as well as the no-click help.

4/05/92 Brian K. Hughes (from Hugh Diedrichs)

POLYEDIT.SC

Used the kernel call DrawPic when erasing lines at the end of the session instead of the room's drawPic method, and added code to redraw overlays and addToPics.

4/03/92 Brian K. Hughes

ACTOR.SC

Move the invocation of (scaler doit:) in Actor's doit method to after the invocation of (mover doit:). This prevents ego from "jumping" or vibrating when he walks and scales.

4/03/92 Brian K. Hughes

SAVE.SC

Adjusted the positions of the YES and NO buttons on the save game delete confirmation Print.

4/02/92 Brian K. Hughes

TALKER.SC

Added color & back properties to Narrator. These properties are used with a clone of the game's systemWindow for displaying talker messages. Note that custom windows (those of type $80) will NOT display the talker's colors.

4/02/92 Brian K. Hughes

ICONBAR.SC

The while loop in the doit method now checks that the state is IB_ACTIVE first, before generating an event. This ensures that, should the state be changed during the process which forces us out of the while loop, we don't get a dynamic event hanging around.

4/01/92 Brian K. Hughes

PRINT.SC

Altered the addText and addButton methods to make use of the new MsgSize kernel call (see below). This eliminates the need for the 1000-byte temporary buffers allocated off heap.

4/01/92 Mark Wilden

KERNEL.SH

Added Message MsgSize function. It is used like this:

```
  (= msgSize (Message MsgSize module noun verb case sequence))
```

and returns the size of the buffer needed to hold the message including the trailing null. Therefore, a blank message will return 1. If the message is not found, the function returns 0. Stage directions, which are normally stripped out when a message is retrieved, are included in the length.

4/01/92 Brian K. Hughes

TIMER.SC

The set60ths method has been changed to setTicks and the real-time calculation in this method has been changed to the correct algorithm. In addition, the parameters for this method have been reversed to make them more intuitive. The format is:

```
      (myTimer setTicks: tickValue who2Cue)
```

The ticks will now compensate for tickOffset, set when restoring a game.

3/31/92 Brian K. Hughes

PRINT.SC

In the showSelf method of Print, made the dialog center itself un-conditionally. Then set the x and/or y according to the x and y properties of Print. This insures that if you pass -1 for either coordinate, that coordinate will remain the value produced by centering the window.

3/30/92 Brian K. Hughes (from Robert Lindsley)

INTRFACE.SC

If a dialog had the time property set, the check method was disposing of the dialog but not breaking out of the while loop in the doit method. Check method now just returns TRUE if the dialog is out of time, and lets Print dispose of the dialog normally.

3/30/92 Brian K. Hughes

ACTOR.SC

Made sure to reset the scaler property to 0 in Prop's setScale method, in case we replace a scaler object with simply y-coordinate scaling.

3/27/92 Brian K. Hughes

ACTOR.SC

The setStep method was not setting the xStep and yStep properties if the actor wasn't scaling. If the actor is scaling, the setStep method only sets the origStep property (a packed word: $xxyy), but doesn't set the xStep and yStep because actor's doit does that for us (if scaling).

3/27/92 Brian K. Hughes

ACTOR.SC

Made sure to dispose of a Prop's scaler in the setScale method, even if we're not attaching a new one. Also, set scaler property to 0 when disposing the scaler in Prop's delete.

3/27/92 Brian K. Hughes

ACTOR.SC
SCALER.SC

Changed the setScale method of Prop to clone the scaler class passed and pass the rest of the parameters to the scaler's init method. This was patterned after the setMotion method.

3/26/92 Brian K. Hughes

GAME.SC

Changed the vanishingY property of class Game from -30000 to 0, which works better for scaling.

3/25/92 Brian K. Hughes (from Randy Mac Neill)

INVENT.SC

Code was added to the advanceCurIcon method of Inventory to set the curIcon's signal to TRANSLATOR or (~ TRANSLATOR), as required by whether the curIcon is the helpIconItem or not.

3/25/92 Brian K. Hughes

ICONBAR.SC

The cond in the noClickHelp of IconBar was replaced with code that unconditionally disposes a modeless dialog (if any), then displays the new help string. This fixed several bugs with displaying the help.

3/25/92 Brian K. Hughes

INVENT.SC

Added a clause to the cond in Inventory's doit method that checks the fastCast and passes the event to the list if it exists. This way, talkers that display messages as a result of actions taken on an inventory item will get events instead of the normal Inventory doit (which highlights items & such).

3/25/92 Brian K. Hughes

ACTOR.SC

Added code in Prop's delete method to dispose of the scaler object (if any) attached to the Prop (or Actor...).

3/19/92 Brian K. Hughes

PRINT.SC

The addButton: method was altered to allow the font to be passed to the new button.

3/17/92 Brian K. Hughes

PRINT.SC

The showSelf: method of Print has been changed to use the window property of Print for the dialog window. If the window property is NULL, the systemWindow global will be used.

3/16/92 Brian K. Hughes

ACTOR.SC

Added an error message in the setScale method of Prop in case the Y value passed is less than the curRoom's vanishingY. This caused problems because of the resulting negative numbers.

In the setStep method of Actor the actor's mover is re-inited if it is a member of Motion now, not MoveTo. Since a class is not a member of itself, this bug caused MoveTos attached to a scaling actor never to be re-inited.

3/16/92 Brian K. Hughes

STOPWALK.SC

Added (self doit:) to the init method. This makes certain that the StopWalk gets a chance to change the actor's view/loop before the actor animates once. This bug was noticable in the way actors would be initially drawn in mid-stride, then change to a standing view one cycle afterwards.

3/13/92 Brian K. Hughes

FEATURE.SC

Changed the approachVerbs method to reset the _approachVerbs property to 0 if either no args are passed or the first arg is NULL.

3/11/92 Brian K. Hughes

CONV.SC

Had to save the caller into a temp and do the (super dispose:) before cueing the caller. This prevents starting another Conversation process before the super disposes.

3/10/92 Brian K. Hughes

   DIALOG.SC
   INTRFACE.SC

   Added an optional parameter to the dispose method of DText and DButton. If TRUE, it will be assumed that the text property is a pointer to a near string, and not a memory pointer to allocated heap.

=--

3/10/92 Brian K. Hughes

   SAVE.SC

   Dialogs were converted to use Print.

3/09/92 Brian K. Hughes

   PRINT.SC

   Made sure to reset x & y to -1 and modeless to FALSE.

3/09/92 Brian K. Hughes

   ACTOR.SC

   Made sure that the setStep method allows for -1 in either parameter, which keeps the original value.

3/07/92 Brian K. Hughes

   PRINT.SC

   Added Print to the prints list only if it is modal. Modeless prints do not want to go on the list because the list gets first shot at events, and modeless dialogs are usually handled by other objects who should get the events instead.

3/07/92 Brian K. Hughes

   TALKER.SC

   Made sure all Prints done from Talker & Narrator's display methods are modeless. This insures that events will get passed to the talker through either the fastCast (if talker is not modeless) or the mouse/keyDownHandler (if the talker is modeless).

3/05/92 Brian K. Hughes

   POLYEDIT.SC
   WRITEFTR.SC

   Adjusted all the print buttons & such to be more pleasing to the eye.

3/04/92 Brian K. Hughes

PRINT.SC

Fixed Prints and Printf procedures to not use a clone of Print. Sending a message to (Print new:) was invalidating &rest.

Made first property of Print able to contain an object ID or an object number ('n'th item in the dialog).

Reset the mode, font, width, caller, title, first, and saveCursor properties in the dispose method so that they have the right values next time.

Added saveCursor property - if set TRUE, Print will change cursor to an arrow cursor then restore it to the iconbar's current icon's cursor.

Set the width property to 0 to default. This fixes the bug with TextSize not recognizing newlines & such.

3/03/92 Brian K. Hughes

ACTOR.SC
SCALER.SC
SYSTEM.SH

Removed the 'vm' designation from all scaling defines. Also renamed the 'vm_signal' of actor to 'scaleSignal'.

3/03/92 Brian K. Hughes

SYSTEM.SH
PRINTOBJ.SC

Changed Print procedure to Prints (print string) and added simple Printf procedure, both of which use Print. Renamed PrintObj to Print (including module & define). As things stand now:

```
PrintObj -> Print    = New class
Print    -> Prints   = Procedure to display near strings
Printf   -> Printf   = Procedure to format & display near strings
```

3/03/92 Brian K. Hughes

PRINTOBJ.SC

Doit method now calls doit method of each dialog item (for cycling icons & such). Also, renamed format method to addTextF. Method now adds text automatically after formatting.

3/03/92 Brian K. Hughes (from Randy MacNeill)

MESSAGER.SC

Added a dispose after printing the error message if the talker can't be found in findTalker. This allows the messager to be disposed cleanly and won't leave it's Set hanging around.

3/03/92 Brian K. Hughes

TALKER.SC

Was referencing a parameter not declared in the parameter list in Talker's display method.

3/03/92 Brian K. Hughes (from Kevin Ray)

INSET.SC

Had to reanimate the cast in the hideCast method so that their new z values will be applied.

3/03/92 Brian K. Hughes

INTRFACE.SC

Changed GetInput to use PrintObj.

3/03/92 Brian K. Hughes

PRINTOBJ.SC

Made sure that x and y parameters were set to 0 if not passed to addEdit.

3/02/92 Brian K. Hughes

ACTOR.SC

Changed origXStep and origYStep to just origStep, which is packed. The setStep method will pack the property with the values passed, and the doit method will unpack them for the scaling computations. Also changed the setScale method of Prop to take four additional parameters: frontSize, backSize, frontY, and backY. Now a scaler can be set with:

```
(theActor setScale: Scaler frontSize backSize frontY backY)
```

The setScale method will attach a clone of the scaler object specified to the scaler property of actor.

3/02/92 Brian K. Hughes

[Multiple Files]

Converted use of Print, PrintD, Printf to PrintObj.

2/27/92 Brian K. Hughes

ACTOR.SC

Changed default view property in class View to -1. This way, if the programmer forgets to set the view he or she will receive an error "View 65535 not found" instead of trying to load view 0. CD guys say this will solve lots of interruptions due to resource loading.

2/27/92 Brian K. Hughes

ICONBAR.SC

Did away with the newCursor temp var in the handleEvent method and just set the cursor to the curIcon's cursor, regardless of whether it or the curInvItem has changed. This way, if the cursor property of the curIcon is changed, it will be reflected immediately.

2/27/92 Brian K. Hughes

ACTOR.SC

1) Fixed bug in turning off the scaling (in setScale method). Changed OR-ing signal with (~ vmAutoScale) to AND-ing.

2) Moved the (mover doit:) code in actor's doit method to above the scaling code. Doing the scaling first sometimes caused PMachine errors.

3) Added origXStep and origYStep properties so that the scaling always has a good set from which to calculate the new values.

4) The setScale method no longer takes a multiplier to be applied to the maxScale property. The argument now represents the y coordinate at which the view is 100% of its normal size. The maxScale is now calculated, based on a ratio of this y coordinate to the vanishingY of the room.

5) Changed the minimum xStep that can be calculated (in the setStep method) from 2 to 1.

2/27/92 Brian K. Hughes

CONV.SC

Bug in MessageObj showSelf. If the sequence of a MessageObj is 0 (indicating that we want all sequences) the Message kernel call should look for sequence 1 for purposes of determining if the messages exist or not.

2/27/92 Brian K. Hughes

POLYEDIT.SC

Changed the name of the polygons file from polygons.999 to 999.pol (where '999' represents the picture number).

2/26/92 Brian K. Hughes

INVENT.SC

Replaced all references to the helpStr property of invItems with code to get the message from a message file.

2/26/92 Brian K. Hughes

ICONBAR.SC

1) Had to save the cursor and signal of the current icon item before calling the item's select method cuz during the select process the item might be disposed. For example, the save button in game controls disposes of the

game controls list in it's select method. After that, the save icon is disposed out of memory, so the iconbar's dispatchEvent method can't reference it anymore.

2) Added noun, modNum, and helpVerb properties to IconItem and changed the noClickHelp method to get messages from a message file instead of printing a help string. The helpStr property has been removed.

3) Fixed bug at top of while loop in the noClickHelp method. It appears that a class is NOT a member of itself. Hence, IconBar was NOT a member of IconBar, which caused the event to be localized. This resulted in A) the event not registering as on an icon if it was within the top 10 pixels on the screen, and B) the text box would flash quickly if the event was within 10 pixels below the bottom of the icon.

2/26/92 Brian K. Hughes

SLIDICON.SC

Added (= window 0) to the hide method of GameControls. In case a control item hides the game controls, this will ensure that the window does not try to dispose again when the game controls are hidden later in the doit method.

2/26/92 Brian K. Hughes

INSET.SC

Instead of reanimating the oldCast in the drawInset method, we reanimate the oldCast in the doit method every cycle, unless hideTheCast is TRUE. Slows insets a bit but it's the only way to insure that the images of the old cast don't get wiped out by text boxes & such.

2/26/92 Brian K. Hughes (from Jack Magn,)

PAVOID.SC

The MergeAPoly procedure has been replaced by a MergePoly kernel call.

2/25/92 Brian K. Hughes

TALKER.SC

Moved the super dispose in Talker to below the call to hide. This ensures that the talker is fully hidden before the caller gets cued, preventing the "overlapping" of talkers and confusion of underbits.

2/25/92 Brian K. Hughes

WRITEFTR.SC

Commented the code that uses the selector dialog item to pick approach verbs. Since selector items can't read from a file and VERBS.SH is game-specific, the selector can't adjust itself from game to game.

When DSelector (or a similar class) can read from a text file, this code will be reinstated.

2/25/92 Brian K. Hughes

USER.SC

In the handleEvent method, if the event is a direction event be sure to check if there IS an iconbar before sending messages to it.

2/25/92 Brian K. Hughes

ICONBAR.SC

Removed code that disposed of the temp variable 'event' at the end of the doit method. The dispatchEvent method already disposes of it and clones that were created in the process of the doit method may end up at the same address.

2/24/92 Brian K. Hughes

INSET.SC

Removed code that saved the old value of useSortedFeatures (can't think why it was put in and it caused a bug!) and re-animated the old cast when an inset gets drawn initially. Since insets normally invalidate the picture we don't want to accidentally obliterate any old cast members on the screen unless the programmer has indicated to do so by settting hideTheCast TRUE.

2/24/92 Brian K. Hughes

USER.SC

Made sure to check user controls before passing a walk event to ego (original functionality that was accidentally removed).

2/24/92 Brian K. Hughes

GAME.SC

Removed explicit Load of CURSOR in save & restore methods. Can't load cursor resources anymore with the new color cursors.

2/24/92 Brian K. Hughes

TALKER.SC

Added a showTitle property (boolean) which, if TRUE, will cause the object's name to be displayed in a title box.

2/18/92 Brian K. Hughes

MESSAGER.SC

The oldIconBarState property was getting set even if we were invoked without being previously disposed. This meant that the oldIconBarState could be set to a DISABLED state if the previous message was interrupted by a new message. To prevent this the oldIconBarState is set only if it is 0.

2/15/92 Brian K. Hughes

WRITEFTR.SC

Updated code using the verbs. Also added code to support approach distance and a few other options, such as letting the approachX and approachY default to the x and y of the object, etc.

2/14/92 Brian K. Hughes

MESSAGER.SC

Made sure the fastCast gets disposed of properly if the conversation is killed (pressed ESC or right-mouse) or if we were displaying one message of a sequence only.

2/14/92 Brian K. Hughes

TALKER.SC

Fixed bug wherein the talker's mouth would keep cycling if you click away the message sooner than the talker expected. If the talker isn't the one talking (ticks are -1) then the handleEvent returns FALSE so the next talker gets a shot at the event. Also, check to see if the event is claimed (by previous member of fastCast list), instead of passing the event to the super.

2/13/92 Brian K. Hughes

INVENT.SC

Fixed bug in call to Message in InvItem's doVerb method. Call was passing NULL for sequence instead of 1. (Note that messager will allow a NULL sequence but not the Message kernel call.)

2/13/92 Brian K. Hughes

TALKER.SC

Made Blink run on real time (it was cycle-based before, which didn't work well) and added a property to Talker called blinkSpeed, which is the number of ticks that will be used as the basis for the Blink cycler. The actual delay between blinks is a random range from (.5 * blinkSpeed) to (1.5 * blinkSpeed).

2/12/92 Brian K. Hughes

SYSTEM.SC

Class Collection now has a new method, called isDuplicate, which determines if an object is already in the list. Each of the add methods (add, addToFront, addToEnd, addAfter) invokes this method and only performs the add if the method returns NULL. For Collection and List, this method simply returns FALSE, since we always want to add, but for class Set, this method returns FALSE only if the set does not already contain the object. Hence, class Set now functions correctly for all the add methods.

2/12/92 Brian K. Hughes

MESSAGER.SC

Made sure each member of the talkerList gets its caller set to 0 before being disposed, then dispose with TRUE rather than the talker's disposeWhenDone property. Also, made sure we weren't sending bogus parameters to the talker's say method from the sayNext method. If the sayNext is invoked from our cue method (at the end of a message), we have no parameters to pass.

2/12/92 Brian K. Hughes

ICONBAR.SC

A line of code that was added to IconBar's handleEvent that returned FALSE if user can't control, was removed again. This was done originally to prevent the ability to middle-click (swap icon) if user can't control. That check has been moved into the code for the middle click.

2/10/92 Brian K. Hughes

FOLLOW.SC

Checked to see if, when the client is within distance and waiting, the angle has changed before trying to set a new angle. This prevents constantly setting the client's heading and invoking the looper every cycle, even if the client is just standing there.

2/10/92 Brian K. Hughes

INSET.SC

Added a style property which determines how the picture will be drawn (if a picture is used).

2/10/92 Brian K. Hughes

INTRFACE.SC

Made ESC return 0 from a dialog with no active items, and any other key returns TRUE. That way we can tell (if we want to) whether the ESC was pressed.

2/10/92 Brian K. Hughes

SAVE.SC
GAME.SC
SLIDICON.SC
INVENT.SC
ICONBAR.SC

Removed text and put into message files (where applicable) and converted some far text used in Format statements to near text.

2/07/92 Brian K. Hughes

TALKER.SC

Several changes:

1) Added a true Blink class for the eyes
2) Gave the fastCast a name so it can be inspected easily in the debugger
3) Narrator's startText method now returns the length of the string (useful for Talker's startText method)
4) Set a talker's ticks property to -1 if he is done talking, but did not pull them from the fastCast or theDoits list. This way they still cycle the eyes even when they're not talking.
5) Made sure the mouth and eyes get set back to cel 0 in the dispose

2/06/92 Brian K. Hughes

SYSTEM.SH

Added global textSpeed (94) which talkers will use to determine the length of time their text should remain on screen. Has no function in CD audio. Note: This global can easily be set by a text speed slider.

2/06/92 Brian K. Hughes

SCALER.SC

New class. Scales its client according to a pair of y values and a pair of percentage multipliers for those y values.

2/06/92 Brian K. Hughes

POLYEDIT.SC

Changed SetCursor kernel calls to invocations of theGame's setCursor. Also set the cursor to ARROW_CURSOR explicitly.

2/06/92 Brian K. Hughes

FEATURE.SC

Was setting a temp variable, theVerb, in the approachVerbs method of class Feature which was unnecessary.

2/04/92 Brian K. Hughes

USER.SC

Added method setSpeed to ego, which sets ego's moveSpeed & cycleSpeed.

2/04/92 Brian K. Hughes

GAME.SC

Move second setCursor call in Game's play method out of the complex message. The system global normalCursor was getting evaluated before the message began, making it impossible to initialize it to a value other than ARROW_CURSOR in the game's init method. This explains why games would get an arrow cursor as soon as the player gets control.

Removed egoMoveSpeed property of class Game.

2/04/92 Brian K. Hughes

MESSAGER.SC

Disposing our Set full of talkers on first doit following the end of the last message, not the second doit.

2/04/92 Brian K. Hughes

FEATURE.SC

On recommendation of myself and Mark Hood, backward compatibility is not being preserved. To that effect, the lookStr and description properties have been removed from class Feature. In addition, all references to the invItem parameter of the doVerb method have been removed. Also, the invItem property of CueObj has been removed.

2/04/92 Brian K. Hughes

FEATURE.SC

The approachVerbs method only functions now if there is approachCode AND at least one argument.

2/04/92 Brian K. Hughes

INSET.SC

Made sure the walkHandler is handled correctly.

2/04/92 Brian K. Hughes

INTRFACE.SC

Print now returns 0 if cleared with the ESC key.

2/03/92 Brian K. Hughes

GAME.SC

The modNum property of class Region is now set to the current room if 0.

2/03/92 Brian K. Hughes

ACTOR.SC

Added a setScale method to View class. If FALSE is passed the scaling is disabled for that object. If a positive value is passed it is used as the multiplier for the maxScale property. Also added a setScale method to Prop class that will take a scaler object as a parameter.

7/12/91 Chad Bye

Palette cycling now supports multiple ranges and multiple speeds. The third argument is now how many system ticks to wait between each palette shift. The sign of the third argument still indicates which direction to cycle. You can (and should) pass all ranges in a single call to Palette.

Example:
```
(Palette PALCycle start1 end1 numticks1 start2 end2 numticks2 ...)
```

7/11/91 Mark Wilden

There is a new resource type for messages. Messages are the analog in non-CD games of the CD resources, sync and audio. You will need a messages= line in your WHERE file to find them. You'll use the ME program to create them. You'll use a new kernel call
```
  (GetMessage moduleNum talkerNum msgNum @buffer)
```
to access them.

7/10/91 Chad Bye

Added picture mirroring as another option for DrawPic's showStyle. Options such as mirroring or blackout should be ORed with the showStyle.

Example:
```
(curRoom drawPic: pic (| IRISIN HMIRROR BLACKOUT))
```

You can also now save the current palette and restore it later. The kernel call Palette now has functions PALSave and PALRestore.

Example:
```
(= savePalette (Palette PALSave))
(Palette PALRestore savePalette)
```

PALSave allocates memory to copy the palette into, and PALRestore disposes this memory. Every time a palette is saved, it must either be restored later or disposed using
(Memory MDisposePtr savePalette).

6/13/91 Pablo Ghenis

Subj: Hunk reporting and requirements

Configuration files are now case-insensitive.

All configuration files (where, resource.cfg, etc.) must now include a line of the form:

```
minHunk = nk
```

where n is a number between 1 and 1024. This specifies the minimum amount of hunk (in kilobytes) required to run the game. If your minHunk number is too large you will get a "not enough memory" warning, if it is too small then any machine will appear to have enough, and some will crash "out of hunk".

To help you determine the correct minHunk, there is a new SCI command line switch "U", which by default will create or append to a file called hunk.use to report hunk usage throughout the game as you play it. The report will include ONLY rooms where hunk usage EXCEEDS the specified minHunk value, so if this never happens then nothing will be reported. This is required for all shipping versions (otherwise they might crash on a machine that barely meets the minHunk requirement.) If you want the report to go to a different file you can specify a file name immediately after the "U", eclosed in double quotes. The "u" switch has been enhanced to work the same way. This option is especially useful when running on read-only media such as CD.

Example:

```
G:> scidh -dvu"c:\myres.use"U"c:\myhunk.use" c:\mywhere
```

The recommended procedure is to start with minHunk = 1k and play the game, going straight to the rooms that use the most hunk, then exit. Near the bottom of hunk.use look for the last line that ends in "NEW MAX!", it contains a better value for minHunk. Replace minHunk in your configuration file with this value, delete hunk.use to get a fresh start and run again, until you end up with no hunk.use file after running the game.

6/5/91 Eric Hart

No one should be using the playBed or changeState methods of Sound any longer. The information needed for 'playBed:' is now contained in the sound files, so simply 'play:' will suffice for every sound. changeState: is no longer needed, because there are three new methods to change the state of a sound object:

```
setVol:  newvol
setPri:  newpri
setLoop: newloop
```

No one should be setting the volume, priority, or loop properties directly any longer, but rather you should go through these methods. (thus, there is no need to call a changeState method after you modify these properties)

Another big change is the addition of priority information to sound files. Sounds will now automatically set their own priorities, so no application should be setting them manually. In the rare cases that this default must be changed, priority may be overrided by programmers using the setPri: method. This will lock the sound object to the specified priority, regardless of the settings in the sound files, until the fixed priority is released by saying (sound setPri: -1). In a case where the sound file contains no priority information, the current value of the priority property will be used to play the sound regardless whether the priority is fixed or not.

Sounds may also now be muted using the mute: method which takes a true or false value. Mute: acts just like pause:, however the sound will continue to be parsed inaudibly until the sound is unmuted by a (sound mute: FALSE). So unlike pause, the sound will continue where it should be, not where it left off.

Also a new controller define has been added to system.sh, cMUTE. This provides for muting individual channels of a sound. We used to simply set the volume of channels that we didn't want to play to 0 by using code like:

```
(sound send: 3 cVOLUME 0)
```

But now we may mute them by using:

```
   (sound send: 3 cMUTE TRUE)    (or false to unMute)
```

The advantage of this over using the cVOLUME way is that the musician may change the volume mix of his sound without the application programmer having to change code, and the channels and voices used by the channel being muted will be freed up for other simultaneous sounds to use.

Existing code which uses the playBed and changeState methods, and that manually sets priority will still work for now, but after the products ship that need this compatibility (like KQ5 cd and Jones cd), these methods will be removed.

More detailed information on these new methods may be found in the comments in sound.sc on s:

3/27/91 Mark Wilden

Backquote key brings up debug, in addition to the regular Shift-Shift-Minus.
New diagnostic resource cursors, selected by -cXX command line option, where XX represents the type of cursor.
This is a decimal bitmapped number where the bits are as follows:

```
   Bit 0:    Change cursor to disk on disk reads
   Bit 1:    Move cursor to left on disk reads
   Bit 2:    Change cursor to chip icon on ARM read/writes
   Bit 4:    Move cursor up/down on ARM read/writes
```

If you don't specify a -c command line option, there is no cursor diagnostics (thus not interfering with the game's cursor). If you specify -c with no number following, the cursor will be as it was formerly (under -d): changing based on disk and ARM access.

3/27/91 Mark Wilden
FileSelector added readFiles: method, which must be called separately from init: (which is called at every arrow key!). Added sort property which, if true, sorts the list.

12/06/90 Mark Hood & Mark Wilden
VIEW.999 is displayed at startup, if it exists.

11/30/90 Mark Wilden
DrawStatus can now take two optional arguments: foreground and background color, which default to the usual black on white. Note that these colors are indexes into the palette, so vRED is unlikely to be actually red (it will select the 4th element of the current palette).

10/23/90 Mark Wilden
FileSet was changed to FileSelector. FileSet used to create an array of file names for use in a DSelector. This function is now integrated into a selector. The file was renamed from FILESET.SC to FILESEL.sc.

10/11/90 Mark H.
A new function selector has been added to MemoryInfo.
it is TotalHunk and returns the maximum hunk space that was allocated by sci at startup on a particular machine and configuration. This is to give application programmers the ability to write conditional code based on the size of a certain machine (memory). It is called as: (for example)

```
   (Printf "Total Hunk = %d" (MemoryInfo TotalHunk))
```

and returns the toal hunk in paragraphs. (1 paragraph = 16 bytes).

```
10/10/90 Mark H.
   The MoveCycle class now respects client cycleSpeed.

10/09/90 Mark H.
   The class PCycle has been added to the system to do palette cycling of
   Props. It tries to take the place of having many  different palette
   cyclers by watching the parameter list. If one number is passed it
   acts like a CycleTo cycling from the current palette to the passed palette:
      (aProp setCycle:PCycle 5) cycles from current palette to palette 5.
   If two numbers are passed it cycles through the range specified
   in the proper direction (i.e. first number to second):
      (aProp setCycle:PCycle 5 2) cycles from palette 5 to palette 2
   If three numbers are passed it cycles through this range a number of times:
      (aProp setCycle:PCycle 5 2 4) cycles from palette 5 to palette 2,
      4 times.

   Of course, a caller can be added to any of the above conventions,
   to cue, as any decent Cycler should.


10/01/90 Mark H.
   The kernel call Lock has been added to lock and unlock resources.
   This is to allow the application programmer to decide which
   resources should locked or unlocked. Lock takes three parameters--
      Resource type... VOCAB SCRIPT SOUND VIEW etc.
      Resource ID... number of resource 999 002 etc.
      Lock or UnLock... TRUE or FALSE to Lock or Unlock.
   Be carefull!!! If you lock a resource yourself, it'll be in hunk forever,
   unless you unlock it yourself. Two new functions selectors
   have been added to the Memory kernel call. They are MReadWord
   and MWriteWord for referencing and dereferencing addresses. Again,
   more power, but more responsibility. You are now free to mess up heap
   and hunk as much as you wish.


09/25/90 Mark H.
   Added Oscillate cycler that will cycle a Props cels forward and
   then back a given number of times. Changed MoveCyc to take a local array
   address instead of a bunch of points so that it can handle more points
   and save wods 'o heap. Removed Features default description and changed
   handleEvent method to return FALSE if there is no description. Moved WordAt
   procedure from PolyPath to system.sc since it has been used a lot recently...
   it will, however be migrated to the kernel shortly. Added Erics changes to
   sound that should fix the restore problem. Changed Grooper to not dispose
   oldMover if it is also clients mover. Added global useObstacles that can
   determine whether or not Ego will use a PolyPath if obstacles are present...
   same for an Actors setDirection.


09/21/90 Mark H.
   DPath now does an extra doit when reiniting itself to eliminate the
   pause at the end of each node. Stop updated actors will no longer pop
   through modeless dialogs when changing their update status. Resource
   purging now uses a prioritized purging scheme based on resource type
   instead of just least recently used. This helps to reduce disk thrashing
   when nearing the edge of Hunk space. To go along with this change,
   VOCAB and TEXT type resources are now unlocked. You may notice now that
   a Load will happen when typing a Said or printing text. If you are completely
   finished with a view in a Room you should use an explicit (UnLoad VIEW xxx)
   since VIEW resource type is one of the last to go otherwise.
```

09/06/90 Mark H.
   Actors delete method was changed to ensure that if a View is addToPic'd
   it will be placed on the features list instead of disposed so that all
   of your handleEvent/doVerb 's will work.

09/05/90 Mark H.
   A caller property has been added to theDialog class. This has made
   it possible to allow the Print procedure to take an optional parameter
   after the #dispose: that is who to cue upon disposal of the modelessDialog.
   The most useful example of this would probably be written as follows:

```
      (instance foo of Script
         (method (changeState newState)
            (switch (= state newState)
               (0
                  (Print "This is a 5 second modeless dialog"
                     #time:5
                     #dispose:self
                  )
               )
               (1
                  (Print "I got cued by the dialog itself.")
               )
            )
         ); changeState
      );foo
```

   Remember that the dispose method of Dialog has been rewritten
   to cue caller if present, so other applications may be possible.

08/30/90 Mark H.
   The kernel call Sort has been added to speed up sorted features,
   however it has been written to be a general purpose sort and thus
   is available for your everyday sorting needs. .
      It is used as follows:

```
      (Sort unSortedList sortedList scoringCode)
```

   where the unSortedList is the list to be sorted, sortedList is an
   uninstantiated (empty) list where the sorted list is to be placed
   by the kernel call, and scoringCode is a piece of code that determines
   the criteria for the sort. The scoringCode's doit should return a
   number for each object that would determine its relative place in the
   list. For example if you wished to sort the cast by their height on
   the screen (their y property) the code would look like this:

```
      (instance sortedList of List)
      (instance yScoreCode of Code
         (method (doit theObj)
            (return (theObj y?))
         )
      )
      (Sort cast sortedList yScoreCode)
```

   and sortedList would now contain the cast sorted by their y properties.
   The Sort algorithm is a simple modified bubble sort. Since most of our
   applications will have less than a hundred elements this should be
   close to optimum.

   BE SURE TO DISPOSE OF THE sortedList!!!! If you do not, you will
   definitely get hard to trace memory frags.

08/21/90 Mark H.
Actors setDirection method has been changed to allow ego to turn if an arrow key was hit and he is up against an obstacle. Previously he would ignore the key completely.

08/14/90 Mark H.
Polygons are here! The Polygon class was added to the system to support Larry's Polygon based avoider. The Polygons are created either manually or with Chad Bye's new Polygon Editor, and added to the obstacle list via the Rooms addObstacle method. The mover PolyPath is where the actual avoider stuff is hidden. It works just like a MoveTo only it will find its way around any polygon in the obstacle list. AddToPics will automatically add a suitable Polygon to the obstacle list when the (addToPics doit:) is called if the addToPics ignrAct bit is not set. The Polygons have also found use in Features onMe method. To do this, redefine the Features onMe method to check if a click was in a surrogate Polygon. The kernel call AvoidPath will check if a given x,y is in a polygon if only these three parameters are passed.

Example:
```
   (local polyPts = [10 20 50 30 50 100 20 100 5 50])
   (instance myPoly of Polygon)
   (myPoly points:@polyPts, size:5)
   (instance fooFeat of Feature
      (method (onMe theObj)
         (return (AvoidPath (theObj x?)(theObj y?) myPoly))
      )
   )
```

If this starts to get used alot, let me know and I'll automate the process a little better. i.e. a polygon property of Feature.
Also, the scads of includes that used to slow down your compiles in system.sh are now in-line with a few exceptions. The following files are NOT included automatically now , and if you're using any of the classes or procedures in their corresponding .sc files, you will now need to explicitly include them. A small price to pay for the speed up in my opinion.
The .sh file that will need to be manually included are:

```
   cat.sh      ;; if you're using the Cat class
   demo.sh     ;; if you're using the Demo Demon
   namefind.sh ;; if you're using the NameFind procedure
   count.sh    ;; if you're using the Count procedure
   lastlink.sh ;; if you're using the lastlink procedure
   gotosaid.sh ;; if you're using the gotosaid mechanisms
```

There were also minor changes to Views (a real isNotHidden method) and the Grooper. Unfortunately, this means that a rebuild is necessary if you use the Grooper.

Also, new tools thanks to Doug Oldfield and Mark Wilden and Chad Bye. Doug has made two new cyclers that are in the system for your use. MoveCycle which repositions the client for each cel. (This seems to be the way alot of artists are doing things these days!) This saves the need for a Script that just positions and sets a cel in every state (Hand carved animation). Also RandCycle that just picks cels at random for a specified number of times, or forever. This is useful for mouth animation while talking. Mark Wildon has created a Dialog building tool that can really simplify 90 percent of your everyday dialog needs. It is a procedure called PrintD and is also now in the system. Chad Bye has added an excellent Polygon Editor (and pather for the other avoider) called

```
   pather. To invoke it, simply put (MakePath doit:) in the appropriate place
   in your debug module (or wherever). Documentation for all is coming soon
   (soon is a relative term), but for now, look at the modules for the
   programmers own docs. Sorry this is so long, but I have been delinquent
   at keeping this up to date, and a lot has happened. As always...
   we're looking for new tools, so let me know if you've made something
   that is nice and general purpose and fills a gap in our current library.


08/04/90 Mark Wilden
   Added new FileIO functions fileFindFirst, fileFindNext and fileExists.
   Added Memory function MCopy.

08/02/90 Mark Wilden
   The function selectors for stack usage kernel calls were backwards;
   i.e. (StackUsage MStackSize) returned the size of the processor
   stack instead of the PMachine stack.

07/09/90 Mark Wilden
   IBMKBD.DRV and TANDYKBD.DRV were changed.  Now the only keyboard
   event modifiers are when shift keys (including Ctrl and Alt) are
   pressed.  The state of Ins, Caps Lock, etc. as well as differences
   in certain BIOSs are now ignored.  The new drivers are in I:, S:
   and the tester directory.

07/03/90 Mark H.

   The Feature class now allows the programmer to decide wether or not
   they would like the proximity checks done if a shift or control click
   was detected. The define NOCHECKMOUSE should be or'ed with the verb
   number for the shftClick and contClick properties.
      (i.e. (theActor shftClick: (| NOCHECKMOUSE verbLook))
   Also a bug in the setChecks method was discovered and fixed.
   The Feature writer now also will allow testing of features created
   and also allows editing of Features, Views etc that were already in
   the room.

07/02/90 Mark H.

   Actors code property now gets done even if object is not updating .
   View and PicView can no longer inherit the dispose method of Feature
   since Feature's dispose now deletes it from the features list.

06/26/90 Mark H.

   There have been several additions to the interpreter.
   Memory allocation and deallocation has been added as well as
   List manipulation functions. Heap memory allocation is handled through
   the kernel call Memory and has three enumerated function definitions:
      (= thePointer (Memory MNeedPtr amount)) -
         requests that "amount" bytes will be allocated and the pointer
         to the memory is returned to thePointer. If the memory is not
         available SCI aborts with the all to familiar "Out of Heap Space".

      (= thePointer (Memory MNewPtr amount))
         same as above, but simply returns a null pointer if memory is not
         available.

      (Memory MDisposePtr thePointer)
         Deallocates the memory pointed to by pointer.

   This function should be used carefully and make sure to deallocate all
   the memory that you have allocated. List manipulations can now be sped up
```

with the ListOps kernel call.

The lines of Code below will speed up the list processing by approximately a factor of four.

```
(method (eachElementDo theSelector)
   (ListOps LEachElementDo theSelector &rest)
)
```

Also available as function definitions are LFirstTrue and LAllTrue. Bob has also put the kernel hooks in for the AIPath from the Chris Iden. More about that later... Stay tuned.

06/20/90 Mark Wilden

The File class and kernel calls have been changed and added to. In the kernel there is now just one entry point for all file functions: FileIO. An enumerated value is sent to FileIO to indicate the desired function (see kernel.txt or kernel.sh for these values). Several new file functions were added: read and write, which operate on arbitrary data; seek, which moves the file pointer; and unlink, which deletes a file.
The File class was changed to reflect these modifications; in particular, the read: and write: methods now refer to raw file access--use readString: and writeString: to read or write zero-terminated strings.

06/13/90 Mark H.

Added new cycler that takes care of the out of sync walk cycler with moveSpeeds greater than 1. Its called SyncWalk and it is in Motion.sc. Also the need for the rebuild is that user has a property called verbMessager that lets you define your own verb message numbers. This will make more sense when my documentation is finished.

06/11/90 Mark H.

We have a new system, folks. A rebuild of your game will be necessary if you are using \games\sci\system. After receiving a lot of great input from Bob, Pablo, Corey, Mark W., Chris I., the SQ4 crew and anyone else I forgot to mention, the Feature class has been completely rewritten (again). The documentation is following closely behind, so sometime tomorrow it should be complete. Besides all the stuff added to Feature, User canInput is now a method. This gives you the ability to break on it and also allows redefinition of it to do things with the cursow (or whatever). There is also a setHeading method of Actor that should now always be used. Instead of code like (anActor loop: facingSouth) you should write something like (anActor setHeading: 180). Besides keeping things consistant between an actors heading and his loop, (for those using sorted features), it also fires a looper if one is attached. There is an optional whoCares parameter (like motion and cyclers) that will cue something when the turn is completed. IF ANY BUGS OR INCONSISTANCIES ARE DISCOVERED, PLEASE LET ME KNOW, POST HASTE. If you don't like the way something works don't abandon it!!! Tell me your suggestions and the system will be better for it. There will be much better doco and examples coming soon. For those wanting to get started with the feature writer let me know and I'll step you through.

06/04/90 Mark H.

The IsOffScreen procedure in sight.sc has been changed due to a bug pointed out by Mark W.. It now correctly used the southEdge define instead of the SCRNHIGH define. 0 is now also considered on screen.

05/24/90 Mark H.

Added MoveFwd Class. Just pass a distance you want to move and the Actor will move in the direction he is pointing. As usual, either preload the module or dispose of it when you're done to prevent frags. See module movefwd.sc for usage info.

05/08/90 Corey

Removed initialization of "version" in system.sh, since you can't auto-initialize a global variable to a string pointer. It is now *required* that you have a line such as:

```
(= version {x.yyy____})
```

(or anything else you want to initialize "version" to) in the (init) method of your room 0 instance of Game. Actually, this was always needed, but simply gave you bad Save games before.

Suggested strings are either {x.yyy____} or {x.yyy.zzz} for "incver" compatibility (either can be used for auto-incrementing version number via incver).

05/04/90 Mark Wilden & Corey Cole

Fixed bugs in "Extra" class when minPause, minCycles, or hesitation zero. Also improved behavior of ExtraEndAndBeginLoop (now allows the pauseCel to be the last cel of the loop).

05/03/90 Bob

GetTime SYSDATE returns date since 1980.

04/02/90 Mark H.

The module user.sc has a slight change to fix a bug pointed out by Mark W. and Gary K. If a horizon was non-zero in a room and ego was ignoreHorizon, Then if ego walked above the horizon he could walk east and not change rooms. Also a fix in demo.sc.

03/20/90 Larry & Corinna

The op. code mod accomodates negative numbers now.

03/20/90 Corey

Does not save control map if it's not going to draw into it for stop updated actors.

03/20/90 Larry & Corinna

When get an "Out of heap space" error, if running with the
sci debugging on (-d) it goes into the sci debugger then exits.
If running without debugging on, it just exits (this is in
case a customer runs into this error).

01/18/90 Stuart

All new and some-what improved sound drivers in system directory.
Changed restart controller to return to loop point in sound,
instead of beginning of sound. This change should not effect
game programmers.

01/08/90 Bob

Increased required "minimum hunk" available to 320K from 136K.
This should not affect anyone during development, and is intended
to reduce problems encountered in the field. Please let me know
if this change causes problems.

12/09/89 Bob

Modified GetTime kernel call as follows to provide more functions:

(GetTime) - returns low word of 60th second counter (unchanged)

(GetTime SYSTIME1) OR (GetTime 1) or (GetTime TRUE) are all
equivalent and will return the same value as before, but NOTE
the "1" or "TRUE" are the non-prefered forms although the value
of SYSTIME1 is also 1.

(GetTime SYSTIME2) - returns a 24 hour clock that is limited to
2 seconds of resolution.

(GetTime SYSDATE) - returns the systems concept of the day it is
in a packed YEAR/MONTH/DAY word.

The standard system files have been modified to conform to this new
functionality.

****
For those who have "frozen" systems and NEED new interpreter
(required for those groups using LOGGER), changes were needed
in system.sc, timer.sc, intrface.sc and kernel.sh
****

This change has a great degree of backward compatibility. To ensure
that you have no problems, merely GREP for GetTime in all your
source file and ensure that IF you passed an argument it was == to
1 (ie "1" or TRUE). If this is not the case change each occurrence
and recompile.

follows some code showing how to use each case. Defines for each
function have been placed in KERNEL.SH.

```
; print current 12 hour time as in 01:30:22 (assume PM)
(= tm (GetTime SYSTIME1))
; HHHH|MMMM|MMSS|SSSS
(Printf
   "%02d:%02d:%02d"
   (>> tm 12)
```

```
      (& (>> tm 6) %111111)
      (& tm %111111)
   )
   ; print current 24 hour time as in 13:30:22
   (= tm (GetTime SYSTIME2))
   ; HHHH|HMMM|MMMS|SSSS
   (Printf
      "%02d:%02d:%02d"
      (>> tm 11)
      (& (>> tm 5) %111111)
      (* (& tm %11111) 2)
   )
   ; print current DATE as in 12/09/1989 (only good through 1999)
   ; years value is number of years since 1980
   (= tm (GetTime SYSDATE))
   ; YYYY|YYYM|MMMD|DDDD
   (Printf
      "%02d/%02d/19%02d"
      (& (>> tm 5) %1111)
      (& tm %11111)
      (+ 80 (>> tm 9))  ; you must add 80 to return value
   )
```

12/01/89 Stuart:

Minor changes to INSTALL.HLP -
1. A warning label for the MT-32
2. Changes in the text of Casio help screens

11/17/89 Jeff:

There is now a "fragment analyser" built into the debugger. It is invoked by 'F' and displays all allocated heap following the first free block. Objects are displayed as usual. Other allocated memory has a code inside angle brackets, followed by the address. The codes are:

<l>	a list
<n>	a list node
<e>	an event
<mxxx>	module node for module xxx
<sxxx>	sound node for sound xxx

If people don't find this particularly useful, let me know. We don't have the code space to make it more useful, and I'd like to pull it out (reducing interpreter size) if it's not being used.

The SC compiler has a new switch (-w) which causes words in the output to be written in Motorola (high-byte/low-byte) rather than Intel (low-byte/high-byte) order. This should lead to faster 68K interpreters, at the price of recompiling the entire game & system source for the 68K. Files affected are all script.* files, vocab.996 (the class table), and vocab.994 (the property offset table).

11/16/89 Corey:
Added "(DisposeScript LOADMANY)" as the last line of the LoadMany function so that it cleans up after itself. Games that were already explicitly doing this (or using LOADMANY as the last argument to the LoadMany function) should probably remove said code.

11/15/89 Stuart:
Wow! Three changes in a row.

MT540.DRV and CSM1.DRV have been revised to correct the installation text. The CT-460 is the same as a CSM-1, not an MT-540.

11/13/89 Stuart:
Optimized MT32.DRV for master volume changes, reverb mode changes, and MIDI channel reassignments. Should have no effect on previous games.

11/1/89 Stuart:
Changes have been made to the following drivers over the last few weeks:

JR.DRV and TANDY.DRV
Enhanced to allow musicians more accurate control over when a song is to start.
ADL.DRV
Enhanced to respond to hold pedal MIDI data.
Adjusted instrument voice volume levels to be more accurate.
MT540.DRV and CSM1.DRV
Fixed a bug that prevented SOUND_DONE cue from happening when the sound was off.

NOTICE TO ALL GAME PROGRAMMERS:
While a game is in development, it is HIGHLY advisable to use G:\GAMES\SCI\SYSTEM for your music drivers. By isolating the current music drivers, the chance of sending Q.A. obsolete drivers is increased. Also, space on the network will continue to be a factor.

F.Y.I - There are currently 32 various ADL.DRV's on the network.

10/24/89 Corey:

Fixed a bug in Print that caused various weird bugs when using modeless dialogs.

10/13/89 Jeff:

The isStopped: method of Actor has been fixed so that it works properly. You should now be able to vary cycleSpeed and moveSpeed independently for Actors on a Walk and have it work properly.

9/22/89 Pablo: GOTOSAID: The end of "not close enough".

TurnIfSaid is a new version of Mark Hood's procedure for making ego face whatever was mentioned. For example if the user types "look chair" ego will turn in place to face the chair before getting a response.

GoToIfSaid is similar but, as the name implies, it moves ego to a specified point or object vicinity before reprocessing the Said event.

These procedures are only useful if sortedFeatures are in use. Grooper and avoider are used to turn and move.

* This takes up ZERO bytes if you don't use it.

9/22/89 Pablo: AVOIDER: Great taste, less filling!

* 28 bytes SMALLER!

* Improved ability to get around convex obstacles (the most common kind).
* No longer tries to reach illegal destinations. This means that you can leave an avoider on ego without it trying forever to get inside a table if the user happened to click there. In such cases you get the same behavior as if the avoider weren't even attached. The same is true for cursor key presses: since off-screen destinations are not in the avoider's department it leaves it all up to the mover, as it should!

* This takes up ZERO bytes if you don't use it.

9/22/89 Mark Hood: GROOPER: Great taste, less filling!

* 112 bytes SMALLER!
* More functional, restores arbitrary old loopers.
* Now works correctly with avoiders.

* This takes up ZERO bytes if you don't use it.

9/19/89 Jeff

Joystick related stuff:
There is a new kernel call named 'Joystick' to do things to the joystick. At present, it only has one function, 'JoyRepeat', which sets the joystick repeat rate. Usage is

```
(Joystick JoyRepeat n)
```

where n may have the following values:

```
-1    Just return the current repeat rate without changing it.
0     Turn off joystick repeat.  In this case, the joystick will
      only return a direction event when the direction is changed.
n     Have the joystick return a direction event (except for the
      'stopped' direction, 0) every 'n' system ticks (which occur
      at 1/60th of a second).
```

The default repeat rate is 0, i.e. the joystick only reports direction changes. For selecting items with the joystick (as in the MenuBar), setting the repeat rate to 30 (two direction events per second) gives a controllable repeat for selection.

The JoyRepeat function always returns the previous value of the repeat rate. Remember to set it back to this value when you're done!

```
(= oldValue (Joystick JoyRepeat 30))
... some user selection ...
(Joystick JoyRepeat oldValue)
```

9/4/89   Pablo

GRAMMAR and VOCABASE change.

The bad news: the words "to", "from" and "about" were being improperly handled, forcing programmers into some kludges to solve the problem. The most infamous example was the need to use TWO said specs to trap the sentences "ask about beer" and "ask bartender about beer"; ie. 'ask/beer<about' and 'ask//beer<about' (yeech!). Also, to and from were sometimes modifiers to the verb and sometimes to the indirect object, adding to the hit-and-miss nature of said-spec writing.

The good news: This is fixed, so 'ask//beer' is enough to trap ALL questions about beer, and to/from are always modifiers to the VERB. Life is simpler...

The price: You will want to grep through your code looking for 'ask/beer<about' and 'give/gold/dwarf<TO' style cases; and change them. My initial grep uncovered less than a dozen such cases per project, so it's not so bad. If you were covering all bases by writing all possible specs then the changes are largely optional. You will need to rebuild regardless, since word numbers have changed...

8/29/89 Corey

Added "track" method to Cat class. Allows Cat to be init'ed after the appropriate mouse event has already occurred, and for cursor keys emulating mouse. (External module invokes "track", which is equivalent to an existing Cat object receiving a mouse down.)
Ask Pablo or me for details.

8/28/89 Pablo

User now has a "curEvent" property which points to the last event processed by User. This is especially useful for following the mouse.

Demo Demon now has FakeMouseUp as well as FakeMouseDown, both of which are really defines that use FakeMouse, which is now tied into User curEvent.

Cat changed to track (User curEvent?) in its doit unless doCast is TRUE in which case it hogs processing. This is a safer implementation for allowing background animation while tracking. Also helps programmatic control of Cats.

8/28/89 Pablo

Class Cat now accepts a caller that gets cue'd when the mouse is released.

Class Track of Motion has been added.

```
;; keep client at a certain x and y offset relative to position of
;; object who
;;
;; client should come AFTER "who" in the cast
;;
;; Usage:
;;    (theTracker setMotion:
;;          Track theTrackee xOffset yOffset zOffset theCaller)
;;
```

8/11/89 Jeff

Buttons other than the left button on multi-button mice are now supported:

```
right button   ->    shift-click of left button
center button  ->    ctrl-click of left button
```

8/10/89 Pablo

QSOUND: Queued sounds now respond to the new sound protocol for cues, where 1-127 are for ad-hoc cues (as before) and 128-64k are for sequential cues. Sequential and ad-hoc cues may be mixed within a song, although the check method of the QueuedSound instance would have to be specialized. This change is backward-compatible.

CAT: Is now a subclass of Actor instead of View, fixing a conceptual bug in the first implementation. This will require you to rebuild.

8/7/89   Pablo

New files FORCOUNT and LASTLINK

```
;; Usage:
;;       (LastLink #client thisScript)
;;       (LastLink #script thisActor)
;;       etc...

;; ForwardCounter Cycle Class
;; Saves states in scripts by cycleing a given number of times
;; then cueing on completion.
;; Usage : propName setCycle:ForwardCounter numOfCycles whoCares
```

8/3/89   Corey

Corrected some of the comments in the StopWalk class. There was an error in the StopWalk usage originally described here. Correct usage example:

```
(actor  setCycle: StopWalk stoppedView)
```

The "walking view" is determined from actor's walking view when StopWalk invoked, not passed as an argument.

7/28/89 Bob

The program, INSTAL.EXE has been modified to adapt to games that do not wish to support a joystick. If the program finds a file named JOYSTICK.DRV on the disk, it will display the use options as normally. If this file is NOT present, the use dialog will NOT be presented.
PLEASE NOTE: This represents a significant deviation from standard Sierra policy, so be sure to inform Garuka of your decision, so that it can be coordinated with documentation and QA.

7/21/89 Bob

OOPS!
On the 14 of June of the year 1989 I, Robert E. Heitman, made a change to the priority band placement in SCI.
The absolute Y coordinate of every other priority band shifted down one pixel. This has given rise to one problem in an older game, where a stop updated door was placed directly on a priority band in order to minimize the size

of the control block.
If you have experienced un-reconcilable priority problems
with existing artwork that predated this change, please contact
me.

7/20/89 Jeff

A new compiler/SCI combination now lets SCI know whether to load the
text.xxx resource when it loads script.xxx. This should speed up disk
access in general when loading a far-textless script, and prevents disk
access when accessing a far-textless script which is in hunk. You should
rebuild your game as soon as possible. If you can't do so right now, the
old interpreter is in x:sciold.exe and x:scivold.exe. The old interpreter
will run files compiled with the new compiler just fine...

7/20/89 Bob

A minor flaw in the debugger has been fixed. The problem related
to "crashing" when you bring up the debugger. The crash could
be manifested in many ways, the most common was "packhandles failure".
Bringing up the debugger could have also caused delayed memory
failures that are hard to pinpoint. Please let me know if things
appear to be less fragile memory wise.

7/18/89 Pablo

QSCRIPT: fixed a couple of bugs in script overlaying mechanism, looks
much more solid now. Required minor bug fixes in game.sc and actor.sc as
well. There is no need to rebuild.

LOADMANY: New procedure LoadMany allows you to load or unload several
resources with a single line of code.

Loading examples:
```
   (LoadMany VIEW 123 232 433)
   (LoadMany SCRIPT 123 232 433)
```

Unloading example (performs DisposeScripts):
```
   (LoadMany FALSE AVOIDER REVERSE TIMER)
```

7/13/89 Pablo

INTRFACE: New method check added to Dialog, fixes bug that kept
modelessDialogs from going away upon expiration of seconds.

7/12/89 Pablo

```
(class QueuedSound kindof Sound

   (properties name "QSnd")

;;Author: Pablo Ghenis, 7/12/89
;;
;;A QueuedSound assumes that the absolute values of the cues it receives
;;form a sequence, ie. 1,2,3... QueuedSounds eliminate the risk off having
;;the animation loop overrun by rapid sound cues since it catches up by
;;cueing its client as many times as the latest increment in signal, thus
;;faking a "queue of cues" (sorry, I can't resist a pun!)

;;size=160 bytes, should save its own weight in ad-hoc code
```

```
   )


7/10/89  Pablo

   INTRFACE has been modified to avoid loading TIMER, a 650-byte savings for
   users of the demo demon or anyone using #time in a Print. Since this
   involves some new properties it will require a rebuild.

   New class TimedCue in script TIMEDCUE. This is a very small substitute
   for timers that will provide a cue after a specified number of seconds or
   cycles; it is a subclass of Script and takes 150 bytes or so. What you
   give up for the 500 bytes is the ability to specify game time, which is
   more appropriate in many situations. No free lunch, but it may be useful.


6/30/89  Pablo

   DPath = D(yanamic)Path and RelDPath motion classes

   This is an alternative to Path, which requires allocation of a static
   array and specialization of the "at" method. D(ynamic)PATH uses a
   dynamically created list to keep path points.

   RelDPath interprets its coordinate pairs as relative instead of absolute
   targets.

   Usage is like other motion classes:

      (anActor setMotion DPath x1 y1 x2 y2 ...  anOptionalCaller)
      (anActor setMotion RelDPath x1 y1 x2 y2 ...  anOptionalCaller)


6/28/89  Pablo

   SCIP Parse tree viewer

   To figure out what said spec to write for a sentence you can now run SCIP
   and type it in. SCIP will display the parse tree in a form that is easily
   translatable to said-ese.

   Example:

      1- type "get big rock" where get is a noun, big is an adjective and
         rock is a noun.

      2- SCIP echoes (the indentation is very significant)
         (Root
            (Root
               (Root .w123))     ;get is root of Root
            (DObj
               (< .w456)         ;big is modifier of DObj
               (Root .w789)))    ;rock is root of DObj

      3- Said spec template is 'Root<modifier/DObj<modifier/IObj<modifier'
         so this case is 'get/rock<big'.

   Use SCIP next time your favorite author demands the ability to parse
   "how do squrrels get into trees?"

6/28/89  Corey

   * Added StopWalk class (STOPWALK.SC).  Use in place of Walk if you
```

```
      have an alternate view to use when ego is stopped/blocked.
      Behaves just like Walk otherwise.  Should be compatible with
      GROOPER, but not with SMOOPER.
;;;
;;; Usage example:
;;;   (actor setCycle: StopWalk walkingView stoppedView)
```

6/28/89 Pablo

* classes Ego, MouseDownHandler and Script updated.
* new file QSCRIPT
* new versions of SIGHT, TEXTRA, SMOOPER, GROOPER

Ego: fixed bug that caused ego to respond to mouseDowns even when (User controls) was 0.

MOUSER: The MouseDownHandler class now has a property name shiftParser which should point to an instance of Code whose doit method will be responsible for putting together a string to be parsed. This opens up the architecture for translating shift-clicks into saidEvents.

```
;; Usage:
;;
;; (instance MyMouseSays of Code
;;    (method (doit what event)
;;       (Parse {look} event) ;or (Parse (what name) event)
;;    )
;; )
;; (instance MyMouseDownHandler of MouseDownHandler)
;;
;; (instance FooQuest of Game
;;    ...
;;    (method (init)
;;       ...
;;       ((= mouseDownHandler MyMouseDownHandler)
;;          shiftParser: MyMouseSays
;;          ,add: cast features
;;       )
;;       ...
;;    );init
;;    ...
;; );FooQuest
```

The Script class now has a "next" property which allows chaining of scripts.

```
;; Usage:
;;
;; (QueScript anObj scriptToChainTo optionalWhoToCallWhenDone optRegister)
;;
;; this will trigger the following when anObj is done executing its current
;; script if any:
;; (anObj setScript scriptToChainTo optionalWhoToCallWhenDone optRegister)
;;
;; scriptToChainTo can be the number of a module which has a real script
;; as public entry zero. This mechanism provides an easy route for
;; "overlaying" portions of long or mutually exclusive scripts to save
;; memory.
```

6/22/89 Pablo

SCPP, the pretty indenter for SCI, now treats comment lines as follows:

```
;;; triple semicolon comments are left-justified
(instance commentExample of FooBar
   ;;double semicolon comments are indented just like code
   (properties
      foo   BAR   ;this is a comment
                  ;and this is a single-; continuation
                  ;which SCPP leaves right where it is
                  ;hopefully aligned with a previous end-of-line comment
   )
)
```

6/9/89   Bob

Certain masochists requested the ability to have alternate palettes contained within the same picture. I have acceeded to their demands. The palettes are specified in PE version 6.000 or higher by pressing 'P' at the color selection window. All palettes are thereafter saved & loaded with that picture.

An SCI programmer can request one of those palettes (0 - 3) as yet another parameter to the DrawPic kernel routine. She may also access an alternate palette by setting the global variable "currentPalette" to the proper value BEFORE drawing the picture via the drawPic method of the room. (NOTE - This method is called in the Room's super init: method)
The full DrawPic protocol:

```
   (DrawPic
      picNumber   ; pic.xxx etc.
      showStyle   ; how to show it
      clearPic    ; TRUE will clear pic, FALSE will overlay
      palette     ; 0 - 3 No error if palette not in data.
   )
```

6/7/89   Bob

I hate writing in this file, so...
If you want an Icon to cycle in a Print window, you need merely tell Print about it using the same tired old #icon: parameter. This time, however, instead of passing view, loop, and cel numbers, you pass an objectID of class DCicon. The class DCIcon is in a file called DCICON.SC, and should be consulted for any further guidance. The following code snippet will do something.

```
   (instance myIcon of DCIcon
      (properties
         view: 120
         loop: 5
         cel: 0
         cycleSpeed: 6
      )
      ;; the init method for the class
      ;; attaches a cycler of Class Forward
      ;; to attach a different class, you should
      ;; instantiate the init method as shown here
      (method (init)
         ((= cycler (EndLoop new:)) init: self)
      )
   )

   ;; please note that there is no way currently
   ;; to produce a cue from the completion of this cycling.
   ;; If it becomes apparent that a need exists, then some
   ;; effort will be made in that direction.

   .
   .
   your code goes here
   .
   .
   ; to use it
   (Print "This Icon will be cycling forward constantly."
      #icon: myIcon
   )
   ; to not use it
   (Print "This Icon will NOT cycle."
      #icon: 125 5 0
   )
```

The Chris squared (MURDER) has working examples of cycling icons.
Please go bother them if this stuff is not clear.

As always, "Write when you get me to do some work."

6/1/89   Bob

GReAnimate has been added to KGraph functions.
It's used as follows:

```
   (Graph GReAnimate top left bottom right)
```

This will have the same affect as a ShowBits, BUT will re-animate
the cast members that are inside the shown rectangle.

6/1/89   Pablo

MORE HEAP!!! The following classes have been taken out of motion.sc and
system.sc and placed in their own files: reverse, chase, follow, wander,
timer and timeout. This saves 1700 bytes (!!!) in rooms that don't use
these classes.

MODULARITY is the word of the day, if you find other system code that
should be in a file of its own please bring it up, everyone may benefit.

The DEMO DEMON is here! To run a game in demo mode, set global variable
demoScripts to some convenient offset, ie. 400. Then for each room to be
demo'ed create a script of actions, make it public entry 0 in its file
and make the script number equal to the room number plus the value of
demoScripts. The following procedures are provided to fake out user
interactions: FakeInput, FakeDir, FakeKey and FakeMouseDown. The demo
demon locks out the real keyboard and mouse to protect your demo from
klutzy showgoers. Overhead is about 380 bytes plus the size of the demo
code, which can range from 100 to 500 bytes typically. Did I mention that
the point of this is to be able to demo your actual UNMODIFIED game?

For internal details see file DEMO.SC, for examples see
\games\ice\source\demo*.sc (especially demo011.sc).

If you need to refer to specific objects in you demo you can use procedure NameFind. Example (NameFind {agent} cast) will return the ID of an object with name property "agent" if it is a member of list cast, 0 otherwise. NameFind can also be used as a poor substitute for public objects.

New and useful procedure added to system.sc: OneOf
Example:
(OneOf 23 10 20 23 30) returns TRUE because 23 (the first arg) IS one of 10, 20, 23 or 30
Also legal is (OneOf client badGuy worseGuy)

CATS: A Cat is an object that follows the mouse, ie. a mouse-draggable gadget. It can be confined to a rectangle or even to one of a rectangle's diagonals, in which case it can be made to track either the mouse x or y coordinate. Animation can be optionally enabled during dragging, although the default of disabling it provides smoother interaction. A cat can be made to affect other objects or variables by customizing its posn method. For examples see \games\ice\source\rm027.sc.

The setLoop method of Actor now also accepts an object as an argument, in which case it will set the actor's looper property and invoke its init just like setMotion does for movers.

GROOPERs a.k.a. GradualLoopers are here.
example: (ego setLoop GradualLooper) will cause ego to go through all intermediate loops when making turns, ie. no more 180 degree "flips". it also works for eight loop views, which can be used for even smoother animation. From system.sh:

```
      ;Standard loop order for actors
      (enum
         loopE
         loopW
         loopS
         loopN

         loopSE   ;new, for 8-loop actors
         loopSW
         loopNE
         loopNW
      )
```

SMOOPERs a.k.a. SmoothLoopers are also here
use same as groopers. Smoopers allow an actor to 'cut away' to a transition view when performing turns, this is especially nice for long

actors. See comments in smooper.sc for details, teleport to Iceman room 56 for live example.

```
------------------
that's all folks! --Pablo
------------------
```

5/10/89 Corey:

You can now do room transitions and picture drawing with no special effects. This is the new default option for picture drawing style. Simply set your room "style" property to PLAIN (defined in system.sh). This is the fastest way to draw a new picture.

5/4/89   Pablo: VCPP

The VCPP vocabulary preprocessor has been enhanced so that the following lines are equivalent:

```
(#synonyms foo bar)
(#syns foo bar)
(#syn foo bar)
(foo bar)
```

The final form is obviously the most concise and is recommended for foreign language vocabulary extensions.

VCPP's internal stack size has been increased in hopes of eliminating occasional reported crashes.

4/19/89 Pablo

To take advantage of heading, a new script SIGHT has been created to provide procedure (CantBeSeen theSight theSeer theAngle theDepth), which tells use whether theSight is within theAngle from theSeer's heading and within theDepth distance. theSeer defaults to ego, theAngle defaults to the right thing based on egoBlindSpot and theDepth defaults to INFINITY, which is defined in system.sh as a very large number.

SortedFeatures: said events now go to all members of cast and features, the burden is on the programmer to use procedures IsOffScreen and CantBeSeen to discriminate an objects responses. Doing so provides a lot more flexibility in said event handling.

INTRFACE has a new procedure called MousedOn which tells whether an event took place within an object's rectangle.

New file MOUSER provides a class of EventHandler that on a simple click can move ego towards the click and then pass a copy of the mouse event to the clicked object if any.

New file RFEATURES provides classes RFeature and RPicView that add properties nsLeft, etc to Feature and PicView respectively. A must if using MousedOn.

New file TEXTRA provides a class TalkingExtra of Extra that defers to a surrogate feature or rfeature if addToPic'd, so that the Extra's responses are guaranteed to be around even on a slow machine.

Class Ego has been moved to USER. User has been simplified so that events are handed to the menu bar, then game, then ego; except for saidEvents which go to menu, sortedFeatures, then game. keyDown events no longer go automatically to the cast, which should speed things up. If anyone needs keyDowns to be sent to the cast (?) then they should add a line to their game's handleEvent to do so.

ACTOR setLoop method now accepts a looper as it's argument, in which case it will behave like setMotion, setCycle, etc. If given a number (ie. all past code) it behaves like it always has.

4/4/89   Bob

Lots of stuff:

Windows are an object that can be modified/customized to let YOU the programmer create that certain look that will sell an additional 100,000 copies of your program.

There is so much to tell and so little time to tell it, but the one thing that every one must do is add the following line as the first statement in your game's init method.

```
   (= systemWindow (SysWindow new))
```

this provides the bare minimum "template" for Print to produce windows that are very similar to what you have had for the last year or so.

I will be working with Mark Hood to produce some the proper documentation that this needs. If you wish, you may read ahead by looking in save.sc for SysWindow and see what properties you may change. For the painfully advanced you may read the full blown defintion of Window in the file window.sc.

As always, "Write, when you get it to work!"

3/13/89 Jeff, Bob, and Stuart

Two new kernel functions allow you to determine something about the hardware on which you're operating.:

```
   (DoSound NumVoices)
```

returns the number of voices in the sound hardware. This lets you know when you're on a single voice PC speaker, for example, so you can only play a song once.

```
   (Graph GDetect)
```

returns the number of colors supported by the video hardware, so you can load a different picture for those mono- and four-color screens.

3/8/89   Bob

The Z property of Actors is now used in Animate to resolve priority/drawing order of objects AND will affect the visual placement of objects on screen. The Z property is a measure of how far UP (above the ground) an Actor is.
For any objects that share the same Y the object with the greatest Z will be drawn last. By the same token for any objects that share the same Y the one with the greatest Z will

appear HIGHER on the screen.

By way of example, consider a car and its door.
The car is at Y = 100, and to APPEAR properly aligned, the door
is at y = 90. This will cause a problem in that Animate will
draw the door and THEN draw the car, erasing the door. Many
solutions exist for this problem, but the PROPER one is to
set the door to y = 100 (same as the car), z = 10. Now the door
will be drawn at the same priority as the car BUT it will be
draw after, so it shows.

As always, "Write, when you get work!"

3/8/89   Bob

All components of a shipping volume based game that USED to reside
in \GAMES\SCI\SYSTEM\INSTALL have been moved up one level to
\GAMES\SCI\SYSTEM. The affected files are:

```
   INSTALL.exe
   space.com
   exists.com
   godir.com
   drvcopy.com
```

An additional file has been added to your shipping disk needs.
This file is called INSTALL.HLP, and contains information on
drivers in a form useable by INSTALL.EXE.

Please ensure that all of your "make disk" batch files are
updated properly.

3/2/89   Pablo

The heading property of Actor has been moved down to Feature, so that now
it is also shared by classes PicView, View, Prop and Extra. This reflects
the fact that things like chairs have a front and a back side, which does
matter when we decide whether to let ego sit down.

*** see 4/19/89 note above ***

2/28/89 Jeff

There are two new motion classes: Path and RelPath. These allow you
to move an Actor along a pre-defined path specified as an array of
points. Using these classes to follow a path requires less memory
than creating a script to do it, and is considerable easier to boot.

You set the motion for an Actor to be a Path or RelPath in the same
way as for other motion classes, but the parameter to setMotion: must
be an instance of one of the classes, rather than the class itself
[for reasons discussed later]:

```
   (ego setMotion: squarePath self intermediate)
```

sets ego's motion to the path 'squarePath' and requests that the caller
be cue:ed when the path is completed. The optional 'intermediate'
parameter specifies an object to be cue:ed at each intermediate endpoint
along the path. The intermediate object's cue: method is passed a
parameter specifying which endpoint caused the cue (first endpoint = 0,
second = 1, etc.).

For both classes of path, two things are required: the array of points, terminated by PATHEND, and an instance of the class with a redefined at: method. The redefined at: method is called with one parameter, n, and returns the nth element of the path array.

The difference between the classes is the interpretation of the points in the array: Path reads them as absolute coordinates to which it should move, RelPath reads them as offsets of the next point relative to the end point of the previous path segment.

To create a square Path (assuming that ego is positioned at 100,100) we do the following:

```
   (local
      sPath =  [
               100   60
               160   60
               160   100
               100   100
               PATHEND
               ]
   )

   (instance squarePath of Path
      (method (at n)
         (return [sPath n])
      )
   )
```

The following will make ego walk the path once:

```
   (ego posn:100 100, setMotion: squarePath)
```

The following uses a RelPath to make ego do loop-the-loops across the screen:

```
   (local
      lPath =  [
               -10   -10
               0     -10
               10    -10
               10    0
               10    10
               0     10
               -10   10
               PATHEND
               ]
   )

   (instance loopPath of Path
      (method (at n)
         (return [lPath n])
      )
   )


   (instance rm1 of Room
      (properties
         picture 1
      )

      (method (init)
         (ego
            posn: 60 100,
            init:,
            setMotion: aPath self
            )
         (super init:)
      )

      (method (cue)
         (ego setMotion: aPath self)
      )
   )
```

2/23/89 Bob

Fixed an obscure bug in menu selection. Bug caused a phantom selection of the first item in the menus that did NOT have a key equivalent, whenever certain keys (ctrl page-up for one) were pressed.
** this change will take effect with 0.000.434 or higher **

2/18/89 Bob

The internal modularity of graphics routines has been shuffled in order to implement VGA graphics smoothly. This change is of most importance to interpreter programmers, but game programmers should be aware that subtle bugs may have been introduced in the process.

2/15/89 Bob

Sounds need not be paused while a menu selection is in process. A new OPTIONAL second argument to the handleEvent of MenuBar will (if present and FALSE) allow sounds to play during the actual selection. If this argument is not present, then things will behave as they have up until now.

NOTE - This will not affect the various (sound pause:) lines that you probably have sprinkled around in your menu code.
I suggest passing the value of (User blocks?) as the second argument to handleEvent AND as the value to all pause: methods.

** this change will take effect with the next new system **

2/14/89 Pablo

New class EventHandler of Set, adds a handleEvent method that passes teh event on to its members and returns as soon as the event is claimed. regions, locales, cast and features are now instances of this class. Also, system-level default handleEvent methods now return TRUE if the event is claimed so, for example, the following code can now be written for an event handler:

```
(method (handleEvent event)
   (if (not (super handleEvent: event))
      (switch  (event type)
         (saidEvent
            (cond
               ((Said 'look')       (Print "you see"))
```

```
                ......
             )
          )
       )
    )
 )
```

2/14/89 Pablo for Bob

New "mapKeyToDir" property of User controls whether key events get mapped to directions.

2/14/89 Pablo, Bob & Jeff

obscure change: on addToPic Views only get init'ed if they are NOT in the cast.

2/13/89 Pablo

Scripts now have properties script and caller, plus a setScript method. This allows one to give sub-scripts to a script, so that scripts can be used in a fashion similar to procedures calling procedures. setScript for all classes that support it now accepts an extra argument to fill the caller property of the script. A script's caller is cue'd when a script is disposed, imitating the behavior of completed motions and cycles. Scripts should self-dispose in their final state to take advantage of this capability.

Also, a third argument is accepted to fill a script's register at init time.

A script's handleEvent passes events to its sub-script.

example:

```
      (instance foo of Script
         (method (changeState newState)
            (switch (= state newState)
               (0 (Print "foo: punting to foo1...")
                  (self setScript foo1 self self)  ;<-- NOTE EXTRA ARGS!
               )
               (1 (Print "foo: back from foo1...")
                  (self dispose)
               )
            )
         )
      )

      (instance foo1 of Script
         (method (changeState newState)
            (switch (= state newState)
               (0 (Printf "foo1: hi\nclient %s\ncaller %s\nreg %s"
                        (client name) (caller  name) (register name)
                  )
                  (= seconds 10)
               )
               (1 (Print "foo1: bye...")
                  (self dispose)             ;<-- SELF-DISPOSING SCRIPT
               )
            )
         )
      )


2/9/89   Bob

   Sometime in the recent past, the OnControl kernel procedure was
   changed to REQUIRE a first argument of which bitmap you were
   interested in examining.  The first argument should be either
   CMAP or PMAP or VMAP.
      The statement (= bits (OnControl x y))

   must be changed to
      The statement (= bits (OnControl CMAP x y))
   to achieve the same results.

      The statement (= bits (OnControl PMAP x y))
   will return the bits of the priority bitmap (similarly for VMAP)


1/27/89 Bob

   The kernel procedure, AddToPic, has been redefined.  It now takes a
   list of PicViews (or sub-classes of PicView) as its sole argument.
   This change will NOT affect programs that do not explicitly re-draw
   the addToPics.  If you do need to re-draw the addToPics, use the
   following line:
      (addToPics doit:)

   If anyone out there is using the old AddToPic kernel routine, please
   see me on possible alternate implementations.

1/26/89 Pablo

   New method for collections (and lists and sets) called "release":
   it deletes all elements from a collection in order to deallocate the list
   nodes. Unlike dispose, it does NOT dispose the elements.

   Used "release" to fix fragmentation caused by use of sorted features in
   User.


1/17/89 Bob

   Hide/Show of stop updated actors should now work much better.
   Please inform me of any problems you may still encounter.


1/5/89 Jeff

   The time at which you receive a cue: upon completion of a motion or cycle
   class has been changed.  It used to be that you would be cue:d at the
   beginning of the animation cycle following the motion completion.  Now
   you are cue:d during the animation cycle in which the motion completes,
   but after the animation is shown on the screen.

   The main game loop looks something like

      (1)   execute doit:s of cast
      (2)   update screen
      (3)   execute Game's doit:
      (4)   execute doit:s of regions
      (5)   execute User's doit:
      (6)   go to 1
```

Before, if a motion were completed in (1), you would not receive the cue: until (1) of the NEXT animation cycle, after all the other doit:s were executed. Now, the cue: comes between steps (2) and (3) of this loop on the SAME cycle in which the motion was completed. This lets you know when the motion completes on the screen in time to set up to do something on the next animation cycle.

1/3/89 Pablo

YET ANOTHER WORD VOCAB FORMAT!
By popular demand I have changed the format accepted by the vocab preprocessor to use a free-form prefix syntax like SCI itself. The vocab.vc file should now consist of arbitrarily nested parenthesized expressions. Each expression should start with one or more preprocessor directives. A directive can be one of the following:

#synonyms:	all words within list have same word number.
#number:	arbitrary jump in numbering.
#<any part of speech>:	all words within list will carry this P.O.S.

In addition, the special word ++ will increment the word number without outputting a record. This should be used to avoid rebuilding the entire game when a word gets deleted from the middle of the word list.

Example vocab.vc:

```
(#noun #number 100
   car
   (#synonyms stone (#verb rock))   ;rock is both a noun and a verb
   ++
   clock
)

(#number 3000)
(#verb
   look
   feel
)
```

For more complete examples, see \games\sci\system\vocabase.vc and \games\ice\source\vocab.vc

To compile your vocab.vc incorporating the basic system word list, type "vcomp vocab.vc".

12/21/88 Pablo

Avoider offScreenOK
   new property, defaults to FALSE, if set to TRUE then the avoider will allow its client to move off screen.

Feature z
   new property, represents elevation of an object. This is important because (+ y z) gives the y of a feature's PROJECTION on the ground.

   The reason we care about this is that if ego is facing south and is in front of a counter, any object on the counter has a y smaller than ego's, so it is computationally BEHIND him. If we take z into account, then (+ y z) can be greater than ego's y, so the object can be correctly considered to be IN FRONT of ego.

Actor's distanceTo has been modified to take z into account. Default z is 0.

12/21/88 Pablo

NEW WORD VOCAB FORMAT
There is a new preprocessor for vocabulary files that allows one to create a file (vocab.vc) in a friendlier format, without word numbers and specifying parts of speech for entire groups of words. There is also a basic vocabulary in \games\sci\system that contains words with parts of speech other than adj, verb or noun, which are game-specific.

New format:

- No parenthesis are used.

- Words with the same word number go on contiguous lines, blank lines cause the word number to increment.

- The default part of speech for all following words can be set with a line like
  ```
          #SPEECH noun
  ```
  This is a vcpp (vocab preprocessor) directive, flagged by an inital #. If a given word following this directive is also a verb then just follow the word with the additional part of speech.

- The current word number can be set with the NUMBER directive, for example
  ```
          #NUMBER 1000
  ```

- Recommended procedure for all games in development:

  Use the new base vocab, define ONLY adjectives, verbs and nouns.
  Use s:bones.vc as a template for your new vocab, starting adjectives at 500, verbs at 1000 and nouns at 2000.
  1-499 and 4000-4095 are word numbers reserved for system use.

INSTRUCTIONS

Create a file named vocab.vc in the new format and copy \games\sci\system\vcompcfg.bat to your vocab directory. Use s:bones.vc as a template and look at s:vocabase.vc for a complete example of new format syntax.

To help you convert your old vocab.txt files to the new format, I have posted two utilities that can be invoked through the following batch files:

vocabare:	strips parenthesis and word numbers from each line
vocwords:	leaves only the words on each line.

If you organize A COPY of your current vocab.txt by parts of speech and insert enough meaningful comments, you can then use one of these utilities to do most of the conversion grunt work. The resulting file should be mush easier to maintain.

Type:
```
      vcomp vocab.vc
```

this will invoke the preprocessor to create a partial vocab.txt, concatenate it to vocabase and hand the result to good old vc.exe to produce vocab.000

If you have any questions I'll be glad to answer them and help out with your vocab conversion.

12/12/88 Pablo

There is a new global variable called "egoBlindSpot" which should contain the size of ego's blind spot in degrees, ie. how many degrees from "straight behind" are not visible to ego. Default is zero (360 degree vision), recommended value is 90 (ego only sees ahead). If useSortedFeatures is TRUE then objects in ego's blind spot do not have their handleEvent method invoked on Said events.

12/1/88
(Jeff) Disk errors (no disk in drive, non-formatted disk in drive, etc.) are now handled better. The user has the option to retry or quit in all cases but save/restore, where he has the option to retry or cancel the save/restore. Let me know if you encounter any odd behavior.

11/23/88

- (Jeff) Added the overlay: method to class Room to overlay the current picture with another. Usage is

```
(curRoom overlay: picNumber [showStyle])
```

  This restores correctly, whereas overlaying with a direct kernel call will not. This supports only one overlay. Let us know if you need more than one.

11/22/88

- (Pablo) Perspective handling has changed to a hopefully cleaner paradigm. Instead of specifying Tilt* properties for rooms, one may now provide values for vanishingX and vanishingY, the coordinates of the picture's vanishing point. The default vanishing point is at 160,30000 to provide the old behavior as a default. By specifying a vanishingY of say -200 one can get more intuitive response to the vertical cursor keys, thus avoiding ego's bumping against walls.

11/21/88

- (Jeff) Initialized global and local variables are now a reality. Some examples of syntax:

```
(local
   foo1                    ;just as before
   [foo2 3]                ;just as before
   foo3  = 1
   foo4  = [1 2 3 4 5]
   [foo5 5] = [1 2 3]
   [foo6 3] = -1
   foo7 = [{a string} {another string}]
   foo8 = 'look/rock'
)
```

  which will generate the following initial values:

```
   foo1     0
   foo2     0
            0
            0
   foo3     1
   foo4     1
            2
            3
            4
            5
   foo5     1
            2
            3
            0
            0
   foo6     -1
            -1
            -1
   foo7     pointer to {a string}
            pointer to {another string}
   foo8     pointer to 'look/rock'
```

Syntax for globals is similar, though as usual you can't declare a global as an array with square brackets.

The script.xxx files generated by the new compiler will be larger than before because they now contain the local variables (which used to be allocated at run time). However, since most variables will be the default of 0, compression will pretty much reduce them to the same size as before for your volumes.

A WORD OF CAUTION: Whenever compiling multiple modules which include your room 0, you must compile your room 0 first. All knowledge of the global variables other than their number is discarded after the first module, so

```
sc rm001 rm000
```

would not generate the global variables in room 0 as it should.

```
sc rm000 rm001
```

would, however, work.

11/19/88

- (Pablo) New AVOIDER:
  A completely new avoider is in, rewritten from the ground up. It now lives in its own file avoider.sc, separate from motion.sc

- Class Motion has two new methods:

```
   setTarget:  updates the x and y properties, doing nothing else
   onTarget:   returns TRUE or FALSE
```

- Class Room has five new properties:

```
   picAngle    0  ;how far from vertical is our view? 0-89
   xTiltTop    0  ;clockwise tilt from x-axis at y-0 (degrees)
   xTiltBottom 0  ;same at y=SCRNHIGH
   yTiltLeft   0  ;clockwise tilt from y-axis at x=0
   yTiltRight  0  ;same at x=SCRNWIDE
```

picAngle is useful to make distanceTo: behave more realistically, and Orbit motions will follow an ellipse instead of a circle for non-zero values

The Tilt* properties completely describe the point of view used to draw a room. Actor's setDirection: has been rewritten to take these values into account so that hitting a cursor key will cause ego to follow the room's apparent directions instead of strict screen coordinates.

To figure out what values to use for these properties, use your eyeball-meter and then trial-and-error to fine-tune. As far as I can tell, typical values are roughly:

```
   picAngle    60
   xTiltTop    0
   xTiltBottom 0
   yTiltLeft   20
   yTiltRight  -20
```

Finding good values for these properties will make it a lot easier for people to play the game without a mouse.

11/16/88

- (Jeff) A relatively major change in the way selectors are handled by the compiler and interpreter has been made and will require a change in syntax of your source code. Fortunately, there is a conversion program which will do the syntax change for you, though you will need to learn a few new habits for writing new code.

  The nature of the change is that there is no longer a distinction between the various forms of a selector, i.e. "state:", "state?", and "state" are identical in all respects except visually. The interpreter now deals with the message

```
   (anObject aSelector [args ...])
```

  in the following way:

```
   (cond
      (- aSelector is a method of anObject
            The method aSelector is invoked with args.
      )
      (- there are no arguments
            The value of the property aSelector is returned.
      )
      (- else
            The value of the property aSelector is set to the
            value of the first argument.
      )
   )
```

  This leads to several niceties:

  - The selector symbol table in sc will now be only a third as large as it is currently, so you'll have a bit more space for your compiles.

  - You can now convert between properties and active values with no source code changes outside the class involved. An 'active value' is a value which has some sort of side effect when accessed. See the document "/games/sci/system/doc/active.doc" for a further explanation.

- System code has been reduced in size by 300 bytes since (for esoteric reasons) we now have 128 more byte-length selectors. Don't be usin' that space, though -- a new Avoider is on the way once Pablo gets well which will eat it all back up.

What this requires in terms of syntax is the following:

- Multiple messages to the same object in an expression must now be delimited by commas, so that the compiler can tell where each message ends. Thus,

  ```
  (ego
     view: 5
     posn: 100 200
     setMotion: MoveTo y x self
     init:
  )
  ```

  becomes

  ```
  (ego
     view: 5,
     posn: 100 200,
     setMotion: MoveTo y x self,
     init:
  )
  ```

- All 'pseudo-selectors' such as 'at:', 'title:', etc. in Print, SetMenu, and other function calls must be preceeded with a '#' to let the compiler know that you're just passing a number rather than trying to access a property. Thus,

  ```
  (Print "This is a pain." at: 100 100)
  ```

  becomes

  ```
  (Print "This is a pain." #at: 100 100)
  ```

The good news is that there is a conversion program, cvtsci.exe, which will do all this syntax conversion for you. Usage is

```
cvtsci file_spec [file_spec ...]
```

You probably just want to 'cvtsci *.sc'. This program can also be run over already converted source, so if you forget to use the new syntax while modifying an already converted file, just cvtsci it again.

Note to the sceptical: the entire source of Leisure Suit Larry II has been converted, compiled with the new compiler, and played to completion on the new interpreter. You really shouldn't encounter problems. If you do, I'm sure you'll let me know.

11/14/88

- If the set:, setReal:, and setCycles: messages are sent to an instance of class Timer, that instance will be used. If they are sent to the Time class itself, the class will create a new: instance and set that instance, returning its ID:

```
(instance myTimer of Timer)
(local
   anotherTimer
)

(myTimer set: 10)                   ;uses myTimer as is
(= anotherTimer (Timer set:10))     ;creates a new: Timer
```

- The stop: and dispose: methods of Sound now take an optional argument which tells the sound whether or not to cue: its client. A value of FALSE means not to cue:. No argument retains the current behavior, which is to cue: the client.

- A fade: method has been added to class Sound. For those with sound systems which have volume controls, this will fade the sound into oblivion, then cue: the Sound's client. For the rest of us, it is equivalent to stop:.

11/11/88

- Class User has acquired 3 new properties. They are:

```
blocks: If TRUE, sounds will be stopped when User is getting input.
x:    Same behavior as X component of Print 'at:' command
   If set to -1 the user input window is centered horizontally
   Any other value specifies the X coord of the left edge.
y:    Same behavior as Y component of Print 'at:' command
   If set to -1 the user input window is centered vertically
   Any other value specifies the Y coord of the top edge.
```

  If your game wants to keep playing sounds while getting input from a window centered horizontaly at the bottom of the screen, send this message to User in your init:

```
(User blocks:FALSE x: -1 y: 160)
```

  (Bob H)

11/03/88

- I haven't documented the new structure of MAKEVOLS yet, so in the meantime, look at \games\pq2\sci\play\resource.txt and puzzle it out. By the by, Friday, Nov 4 is my 37th birthday so don't hassle me.

10/25/88

- Notes on the use of &rest:
  There is a bug in &rest which is not likely to be fixed right away, but there is a fairly simple workaround. The problem arises when &rest is used in an message send with a computed reciever, as in:

```
((expression) arguments &rest)
```

  The above expression will be evaluated as

```
((expression &rest) arguments)
```

  which is clearly not what is intended. The work around is to do the message send in two steps using a variable to contain the computed reciever:

```
(= reciever (expression))
(reciever arguments &rest)
```

10/24/88

- Changes have been made to both the kernel and the compiler to speed up critical sections of kernel code. They should be transparent to game programmers except that you will now need vocab.994 (on s:) in order to run your game.

  NOTE TO SYSTEM PROGRAMMERS: when remaking the system from now on, use the -O option of sc to cause it to write vocab.994. This contains the offsets to certain properties of certain classes, allowing the kernel to look them up much more quickly. The classes and properties which are affected are in "offsets.txt", which must be kept in synch with the enum in "i:selector.h".

10/20/88

- To change pictures within a room, you should use the drawPic: method of Room (curRoom drawPic: 100). This method NOW disposes of any addToPics that have been accumulated. This cures a symptom of restoring a room with the alternate picture up and seeing disembodied views on the screen.

10/19/88

- The SC compiler now has a -a option which aborts the compile immediately if the class database is locked.

- The debugger now has full command-line editting, using the same routines as those used by edit items in dialogs. You can even use your mouse. Also, stepping across sends & calls (using Tab) works again.

10/17/88

- &rest should now work with kernel functions. Let me (Jeff) know if you have any more problems.

- (Printf) now works. Syntax is just like C's function:

  ```
  (Printf formatString [parameter parameter ...])
  ```

  Note that if you want to do anything fancy (like titles, fonts, centering, etc.), you'll still need to use (Print (Format @str ...)). However, for simple print windows (particularly for debugging), Printf works just fine.

10/04/88

- A class browser (a la Smalltalk) is now available as the BRIEF macro "browse.m" in s:. Compile with "cm browse". I believe that it will only compile for BRIEF v2.xx. For those running v1.xx, try changing such things as "(+= foo bar)" to "(= foo (+ foo bar))", etc. Or upgrade to v2.xx!

  To run the browser, assign a key to run the macro "BrowseClasses" or use the "execute macro" function to run the macro. The browser needs an environment variable, "browse", which is the path along which to search for the source code for the classes. You'll probably just want to "set browse=s:". The browser also uses a small file, ro.com (in y: on the net), to determine whether or not a file is read-only. This program, invoked as "ro file", returns a DOS exit code of 1 if the file is read-only, 0 otherwise.

When you invoke the browser, there will be a small delay, then a menu of the system and user-defined classes will appear on the left of your screen. This menu (generated by the new sc compiler in the file "classes") also shows the hierarchical relationships of the classes. Select a class by using the arrow keys, Home, and End or typing in the name of the class you want to inspect (an incremental search is done based on what you've typed). When the highlighted bar is on the class you want, press Enter. If you want to bail out, press Esc or Alt-minus (this applies to all windows in the browser).

A new menu will be displayed which contains the selections "DOCUMENTATION", "PROPERTIES", and the names of any methods which the class defines or redefines. Select an item as above and again press Enter. What happens next depends on what you selected:

PROPERTIES
: A new window pops up showing the properties which the class either defines or whose value it redefines. This is a display-only window -- you can't edit in it.

DOCUMENTATION
: Pops up a window displaying the class heading of the class, which is where we will be documenting the class. This is a quick way of finding out what the class is about. Any suggestions on improving the documentation might be welcome. This is a full editing window, allowing you to scroll through and edit the source file containing the class.

method name
: Pops up a window displaying the selected method. Again, this is where documentation on the method will be kept and the window is a full editing window.

If the source file which is found for the class is read-only (those on s: are), the browser will beep and display the message "File is readonly." to let you not to make any changes which you expect to save.

If you make any changes in the file and then press Esc, you will be asked if you want to save the changes: your options are (a)bort [forget the changes], (w)rite [save the changes], and (c)ontinue [dont' leave the file yet].

One caution: BRIEF claims to only support three pop-up windows at a time. When you're browsing a class you'll have that limit on the screen, so popping up another window (todo list, game.sh, buffer list, etc.) might blow you away. Not knowing the internals of BRIEF (would that I did!), I can't say what will occur.

10/03/88 "Feature" class (Pablo)

- A new class call Feature is now available. A feature is an object with properties x and y and a HandleEvent method. Its purpose is to respond to Said events (user input). Features must be static instances, and get added to the global features list by using a new room method called

SetFeatures. A room's dispose method now disposes of all features, since they are conceptually room-local.

The x and y properties are only used if a procedure called sortedCpy in user.sc is uncommented (in a private copy of course). What this does is cause actors, props and features to get a shot at the said event in order of decreasing proximity to ego, which is more realistic. However, enabling this capability takes up a few hundred bytes of space so I have left it out of the system-wide version of user.sc. See me if you want to enable it.

Example:

Create a simple feature with:

```
(instance Table of Feature
   (properties
      x  100
      y  120
   )
   (method (handleEvent event)

      (cond
         ((or  (event claimed?)  (!= (event type?) saidEvent))
            (return)
         )
         ((Said 'look/table')
            (Print "there is an empty cup on the table")
         )
      )
   )
)
```

and activate it in a room's init method with:

```
(self setFeatures: Table Chair Pot) ;this room has three features
```

10/03/88 TRIGONOMETRIC FUNCIONS (Pablo)

- GetDistance now takes an optional 5th argument, perspective, which is the users point of view of the room in degrees away from the vertical along the y axis. The typical value for Sierra games seems to be about 60 degrees. This argument defaults to zero if absent. What this change does is to make each y-pixel represent a greater distance than an x-pixel, a behavior closer to reality.

  I suggest modifying the distanceTo: method to use a global variable called perspective that could be reset in each room init.

- Four kernel calls have been added to perform integer trigonometry. All four take as args an angle in degrees and a number.

  ```
  (SinMult anAngle aNumber) returns aNumber*sine(anAngle)
  (CosMult anAngle aNumber) returns aNumber*cosine(anAngle)
  (SinDiv anAngle aNumber) returns aNumber/sine(anAngle)
  (CosDiv anAngle aNumber) returns aNumber/cosine(anAngle)
  ```

  CosDiv, for example, is used in the new GetDistance.

  All other trig operations can be built from these four calls.

10/03/88

- Class Game has notify: and setScript: methods and a 'script' property just like Region and Room. Thus, your room 0 can now have a script and communicate with Regions and Rooms.

- The class Locale has been added (in game.sc). A Locale is similar to a Region, but only has a handleEvent: method. It is the object of choice for handling default responses to user input which used to go in Regions (e.g. 'look/meadow', 'look/tree', etc.).

- The source code in s: is now much more heavily commented. We'll try to keep the comments up to date and useful enough to replace the documentation, which just never seems to be current.

09/27/88

- Resource usage tracking is now available with the '-u' option on sci. When resource tracking is enabled, the interpreter will write a line to the file "resource.use" each time a resource is loaded from disk. The line is of the form

  ```
  rmxxx  resType.yyy
  ```

  which indicates that resource type resType, number yyy, was loaded in room xxx. Since the resource cache is flushed on each newRoom: when resource tracking is on, this will indicate which resources each room uses.

  A more useful report than "resource.use" can be generated by

  ```
  sort <resource.use | uniq >uses
  ```

  which prepares a sorted list with duplicate lines removed.

  Note that "resource.use" is APPENDED to each time you run sci with resource tracking on -- it is not overwritten. You will want to delete it periodically to keep it from growing too large or to keep the usage patterns current.

09/26/88

- The order in which various elements of your game get control has been changed. It used to be that control went first to the cast, then the user, then the regions. One sort of problem which was created by this was the following from KQ4:

  - The cast gets control and advances ego to the edge of a cliff where ego is positioned on the control line at the edge.
  - The user gets control next and types 'wear crown'. This is duly parsed and passed to the handleEvent: methods, which decide to start ego's transformation to a frog.
  - Control now passes to the regions, which note that ego is on a control line and attempt to start ego falling.
  - Much confusion on screen.

  The modification just made now passes control first to the cast, then to the regions, then to the user. This should solve problems arising from the scenario above. It might, however, lead to other problems of a timing-related nature which we haven't anticipated. If you suspect that this change has done you in somehow, please talk to one of us -- we may be able to come up with a way around your problem, or the change can be backed out (it only involves swapping two lines of code).

09/25/88

- The 'o' and 'O' commands in the debugger now display all objects in one window in which the new: objects are preceeded with a '*'. This gives you a better feel for where objects are positioned in relation to each other.

- Many changes have been made to the Sound class. The 'keep' property no longer exists -- to stop a sound from playing without disposing it, just invoke the stop: method:

  ```
  (mySound stop:)
  ```

  The dispose: method still exists, and not only stops the sound but disposes its soundNode (an internal kernel structure) and the sound itself if it is a new: object.

  The play: method of Sound now sets the 'loop' property to 1 if it is 0 when play: is invoked. Thus, the expression

  ```
  (mySound loop:1 play:)
  ```

  may now be written

  ```
  (mySound play:)
  ```

  if you can be sure that the loop property is either 0 or 1.

  Sounds now have an 'owner' property, which can either be left empty (= 0) or set to the ID of a Region or Room. When a Region or Room is disposed, the system disposes all sounds with either no owner or which are owned by the Region or Room being disposed.

  Sounds are no longer automatically disposed when they finish, only on room/region changes. Because of this, a sound which might be played many times in a room (e.g. a non-fatal fall or playing an instrument) should be an instance rather than a new: Sound. Since the new: sound will only be disposed on a room change or if you explicitly dispose: it, (Sound new:) can potentially fill the free heap.

  To minimize heap fragmentation of sounds in regions, make them instances of Sound, initialize them in the init: of the Region, and set their owner to the region, e.g.

  ```
  (mySound owner:thisRegion init:)
  ```

  Do not dispose: sounds in regions -- only stop: them. Since dispose: frees several sound structures in the heap, the next time the sound is played it will reallocate them ABOVE the current room, leading to fragmentation when you leave the room.

  The general rules of thumb for sounds are:

  - Always make sounds instances of Sound -- don't use (Sound new:).
  - Never explicitly dispose: a Sound unless you're sure you know what you're doing -- use stop: to turn it off. Sounds will be

automatically disposed on Room/Region changes using their 'owner' property.

09/21/88

- Sci accepts a '-d' command line option to enable debugging. This means that you will get a 'PMachine' error rather than an 'Oops!', and be dropped into the debugger as before. As time marches on, more of the debugging facilities will be added to this switch.

09/18/88

- The debugger is now invoked by pressing the "grey minus" (numeric pad) while both shift keys are down. Also, PMachine errors are reported in a shippable manner UNLESS you have invoked debug via the keyboard prior to encountering the error. This is subject to change, stay tuned for developments.

  NOTE: On the TANDY 1000, the proper minus key is the one also marked DELETE. You must have numlock OFF to access it.

09/16/88

- Prioritized sounds are now a reality. The 'priority' property of Sound is a signed integer which defaults to 0. If you tell a sound to play: and a lower priority sound is already playing, the new sound will interrupt it, play to completion, and then restart the old sound at the point of interruption. If the new sound has a priority equal to or lower than the old sound, it will wait for completion of the old sound before it plays.

  There are three properties of interest for coordinating sounds with each other or with animation.

```
   signal
      This gets set to the value of an animation cue for one animation
      cycle, after which it is reset.
   prevSignal
      This gets the value of signal when signal gets set, but retains it
      until the next sound cue.
   state
      This is the state of the sound, and is one of the following:
         SND_NOTREADY
            The sound has not been initialized.  Default for Sounds.
         SND_READY
            The sound has been initialized, but not submitted for playing.
         SND_BLOCKED
            The sound has been submitted for playing, but either the
            sound is paused or a higher priority sound is presently
            playing.  (Note that since the debugger pauses sounds, an
            active sound will always appear as blocked in the debugger.)
         SND_ACTIVE
            The sound is currently playing.
```

  The changeState: method allows changing the priority or loop count of a sound which has been initialized (or is playing). Just set the appropriate properties of the sound, then do a (mySound changeState:).

  You can use priorities to chain sounds together without the 'next sound' stuff (which will probably be eliminated soon). Just submit the sounds with decreasing priorities:

```
      (firstSound loop:1 priority:2 play:)
      (secondSound loop:2 priority:1 play:)
      (thirdSound loop:1 priority:0 play:)
```

Will play one loop of 'firstSound', two loops of 'secondSound', and one loop of 'thirdSound' in that order.

To terminate a sound at the end of its loop rather than immediately, just set its 'loop' property to 1 and do a changeState:

```
      (mySound loop:1 changeState:)
```

There are times when you will want to play a sound only if no higher priority sound is playing (like the 'got item' music in KQ4). Submitting it at a lower priority will keep it from playing when a higher priority sound is present, but will play the sound when that higher priority sound is finished. What we want is to either play the sound immediately or not at all. To do this, use the 'playMaybe:' method:

```
      (mySound playMaybe:)
```

09/14/88

- The EndGame class and the end method of Game are kaput. Following is an example of how to end your game. This example uses a "dead" global var to detect the need for ending.

```
   ; this may or may not already be in your rm000
   (method (doit)
      (if dead
         (repeat
            (switch
               (Print
                  "\"Thank you for playing\n
                  King's Quest IV,\n
                  `The Perils of Rosella.'\n\n
                  Next time... be more careful!\""
                  icon: 100 0 0
                  mode: teJustCenter
                  title: {Roberta says:}
                  button: {Restore} 1
                  button: {Restart} 2
                  button: {____Quit____} 3
               )
               (1
                  (theGame restore:)
               )
               (2
                  (theGame restart:)
               )
               (3
                  (= quit TRUE)
                  (break)
               )
            )
         )
      else
      ; your normal code
         ...
         ...
         ...
         ...
         ; this must be here
         (super doit:)
      ) ; end of NOT dead
   )  ; end of DOIT:
```

09/07/88

NOTE:
   The two following changes were not arbitrary, and I will be
   willing to justify them to anyone who cares.

```
-  The selectors to the (Display) kernel procedure have changed.
   The following selectors are valid:
      p_at:
      p_mode:
      p_color:
      p_back:
      p_style:
      p_font:
      p_width:
      p_save:
      p_restore:

   NOTE: The colon must be included.

-  The selectors to the (SetMenu/GetMenu) kernel procedures have changed.
   The following selectors are valid:
      p_said:
      p_text:
      p_key:
      p_state:
      p_value:

   NOTE: The colon must be included.
```

- Added code to MenuBar to respect "state" property. If "state is
   FALSE, then handleEvent returns FALSE.

09/06/88

- Added a new property to User. "inputLineAddr" is the address of
   the input line being used by User to get input (sort of circular logic).
   One possible use is as follows:

```
   (Print
      (Format
         "%s, %s...\nBoy! listen to that echo!"
         (User inputLineAddr?) (User inputLineAddr?)
      )
   )
```

09/05/88

- In order to get the sound state to display properly in your menus after
   a restore game, you'll need to make several changes to your code (we'd
   do it for you in the system, but the system doesn't know the details of

your menus).

1) Break the enum for your menu items out into a "menu.sh" file. This has the added advantage of letting you include the file in any module in which you wish to do something to your MenuBar.

2) Include "menu.sh" in the file which includes your instance of Game (rm000, kq4, or whatever).

3) Redefine your Game's 'replay' method as

```
(method (replay)
   (SetMenu soundI
      text:
         (if (DoSound SoundOn)
            {Turn sound off}
         else
            {Turn sound on}
         )
   )
   (super replay:)
)
```

'Replay' is the method which is invoked by the system instead of 'play' when a game is restored. If you need to do any setup on a restore (versus a restart, which invokes 'play'), put it in this method. The (super replay:) must ALWAYS come at the end of 'replay', since it contains the main game loop.

- Ego should be able to leave a room under diagonal motion now. In order to do this, class Ego now ignores the horizon. Let me know if this causes any problems.

09/02/88

- The initialization of the menu bar hase been removed from the (theGame init:) method. Each game must now init it at the proper time. You can achieve the behavior that you're accustomed to by adding the following line to your room 0 right after the call to super init: in your game init:

```
(TheMenuBar init:)
```

If you wish to defer the drawing of the menubar, you should remove the line (self draw:) from your menu init and place the following line where you want the menubar to first become visible:

```
(TheMenuBar draw:)
```

NOTE: an undraw menubar will still respond to user input. ie.. ESC clicks, key strokes and typed input.

- MoveTo has been fixed to not skip over control lines. The fix reduces the effective speed of motion of Actors that fit this general set:

xStep <> yStep - The greater the difference, the greater the affect heading NEAR nw, ne, sw, or se.

This may cause some skating in actors with vastly different step sizes. Please see Bob H. for more details.

- Sound volume is now in. See the volumeI item of s:menu.sc for how
  to implement it.

- Sound volume and sound on/off state should be retained across
  restore/restart. This change requires some change to your menu handling
  of sound. Look at the sound stuff in s:menu.sc to see how to modify it.

  You can find out what the current sound state is by doing a

  ```
     (DoSound SoundOn)
  ```

  Do not use the old 'value' feature of the sound menu to track the state --
  querying the interpreter directly is much safer.

  Volume (0-15) can be queried by

  ```
     (DoSound SoundVolume)
  ```

08/30/88

- ShakeScreen is now supported. The kernel call is as follows:

  ```
  (ShakeScreen n [d])
     n =   number of shakes, 1 to 64K
     d =   direction to shake; 1 = down, 2 = right, 3 = down/right
        default direction is down
  ```

- Save/restore dialogs have been modified. Should handle full-disk
  conditions, disk changes, etc. much better.

08/30/88

- There is a global variable named 'version' which should now point to
  the version string for your game. In the init: of your game, put the
  line

  ```
     (= version {x.yyy.zzz})
  ```

  if you're using incver.exe to maintain your game version number (you
  should be).

  This will allow more stringent checking for validity of games before
  restoring them.

08/29/88

- There is a new basic Sierra menu (menu.sc) on the net. It contains
  the code for echoing the input line (using modifications to User outlined
  below) and for setting animation speed and sound volume (using class
  Gauge outlined below).
  It also contains modifications to the said specs for various items. A lot
  of caution must be exercised in writing menu said specs, since the MenuBar
  gets the first shot at events. For example, using 'save' as the said spec
  for saving a game means that if the user types 'save man' he will get the
  save-game dialog. The said spec should be 'save[/game]'.

- Class User has a new property, 'prompt', and a new method 'getInput'.

  ```
     prompt
  ```

This is the text which will be displayed above the edit box when the user starts typing. Its default is 'Enter input', but it can now be changed (e.g. to 'Give her your best line, Larry').

getInput:char
Collects a line of input from the user and passes it on to the parser. The first character of the input line is 'char' unless 'char' is (User echo?), in which case the previous input line is echoed. The default value of 'echo' is now the spacebar. The use of F3 for echoing is done through the menu, which invokes (User getInput:(User echo?)).

- A new class, Gauge, has been added to the system. A Gauge is a dialog which uses a thermometer-like gauge to allow the user to adjust something (animation speed, sound volume, etc.) A kindof Dialog, it has the following added properties and methods:

  description
  This is the text which appears above the thermometer and tells the user what is being adjusted and how to adjust it. E.g.
  "Use the mouse or right and left arrow keys to
  select the speed at which characters move."

  higher
  This is the text which goes in the button which increases the value of whatever is being adjusted. Its default is "up".

  lower
  This is the text which goes in the button which decreases the value of whatever is being adjusted. Its default is "down".

  minimum
  The minimum allowed value of whatever is being set. Defaults to 0.

  maximum
  The maximum allowed value of whatever is being set. Defaults to 15.

  normal
  This is the value of whatever is being adjusted which you, the game programmer, consider the normal value. Defaults to 7.

  update:
  A method used internally.

  As for Dialogs, the 'text' property is the title of the Gauge. To use the gauge, do

```
(= newValue (Gauge doit:currentValue))
```

  where 'currentValue' is the current value of whatever you want adjusted. The doit: method of Gauge returns the value selected by the user.

  Gauges for animation speed and sound volume are now built into the standard menu.

  A Gauge needs about 1K of heap to run, and can be disposed immediately after use: (DisposeScript GAUGE). The class should be included on all volumes.

- The inventory dialog now supports up- and down- arrows as well as Tab and Shift-Tab.

08/23/88

```
-  save: and restore: methods of Game now prompt for disk changes if
   the current device is the target and is removable.  In support of
   this, the following kernel calls have been added:

-  (StrAt string position [char])
      StrAt returns the character at 'position' in 'string'.  If the optional
      'char' is specified, it replaces the character with 'char', returning
      the old contents.  Use this rather than the mask-and-shift method
      which I expounded to Teresa (sorry...), since byte order in a word
      will vary between machines.

-  (DeviceInfo function @string [@string])
      DeviceInfo returns information about file system devices.
      The functions are:

          (DeviceInfo GetDevice @path @device)
             Puts the string describing the device component of 'path'
             into the string pointed to by 'device'.  Thus, if
             path = "g:/games/kq4/sci", device = "g:".  If there is no
             device component in 'path', puts the current device in 'device'.

          (DeviceInfo CurDevice @device)
             Puts the string describing the current device in 'device'.

          (DeviceInfo SameDevice @dev1 @dev2)
             Returns TRUE if the strings pointed to by 'dev1' and 'dev2' are
             the same physical device, FALSE otherwise.

          (DeviceInfo DevRemovable @device)
             Returns TRUE if 'device' is removable, FALSE otherwise.
```

08/22/88

- Class Extra (see 8/19/88) is now officially part of the system. Like Jump, etc., it takes no overhead unless you use it.

08/19/88

- New stuff has been added to allow semi-automatic adjustment of animation to the speed of a machine. Here's how it works:

  A global variable called 'aniThreshold' contains the animation interval (see 'aniInterval' below) which you consider to be just barely acceptable (the default system value is 10). After the startRoom: method of the Game has been invoked in the process of doing a newRoom:, the checkAni: method is called.

  checkAni: does animation cycles (only drawing objects, not calling their doit: method) and checks aniInterval against aniThreshold. If aniInterval is too large, checkAni: looks for the first member of the cast which is marked as an 'extra object' and does an addToPic: on it. It continues to do so until either aniInterval drops below aniThreshold or it runs out of 'extra objects' in the cast.

To mark an object as 'extra' (not to be confused with the class Extra in development by Al Lowe et. al.), use the isExtra: method:

```
isExtra: TRUE        ;marks the object as an extra
isExtra: FALSE       ;unmarks it
isExtra:             ;returns the current status
```

Things such as sparkles or ripples on water, woodpeckers in trees, etc. could all be dropped out in this manner. Note, though that if an object is added to the picture in this manner, it will no longer be accessible through any object ID you had before. The class Extra (below) is, by default, marked as extra.

- A new class, Extra, developed by Al Lowe is in s:extra.sc. It will be made a part of the system on Monday (which will require a recompile of the world). An Extra is a subclass of Prop which waits for a random time interval, then cycles for a random interval and repeats. It's great for people reading magazines in waiting rooms, woodpeckers pecking trees, etc. It has the following extra properties:

  ```
  pauseCel
     Cel to pause on when not cycling: -1 for random cel, -2 for last cel
  minPause
     Minimum number of animation cycles of no action when paused.
  maxPause
     Maximum number of animation cycles of no action when paused.
  minCycles
     Minimum number of animation cycles of cycling when active.
  maxCycles
     Maximum number of animation cycles of cycling when active.
  ```

  This class is, by default, an 'extra' in the sense used above in regard to animation speed, and so will be added to pic if the machine is too slow.

- 'O' and 'o' work in the debugger again -- the stack size reduction had killed them. In fixing that, I also added code to keep inventory items from being displayed, so the static object display won't be so cluttered.

- The old global variable named 'overRun' has been renamed to 'aniInterval' (animation interval) and now contains the number of timer ticks that it took to do the most recent animation cycle, rather than how much longer than 'speed' it took. This is a much more useful value.

- Icons are now centered in a Print window if there is no accompanying text.

- Edit window length now more closely matches the number of characters which can be typed. Also, the input line and save-game description have been lengthened.

- The parser recognizes modifiers once again.

08/15/88

- Several new PMachine opcodes have been introduced, leading to more space efficient code. The system object code size was reduced by about 1200 bytes.

08/12/88

- Menu stuff, which doesn't seem to have been written up before:

  You can modify an existing menu with the 'SetMenu' kernel command, which has the following syntax:

  ```
     (SetMenu itemName selector value [selector value ...])
  ```

  where itemName is the name by which you refer to your menu item and the the available selectors are

  ```
     said: newSaidSpec       change the said spec for the menu item
     text: newText           change the text displayed in the menu
     key: newKey             change the key which selects the menu item
     state: newState         = dActive to enable menu item
                             = 0       to disable menu item
     value: newValue         change the value to return when selected
  ```

  For example, you can disable the save-game menu item (which we'll assume is refered to by 'saveI') with

  ```
     (SetMenu saveI state:0)
  ```

  The 'GetMenu' kernel function returns the current values of a menu item corresponding to the selectors listed above. Thus,

  ```
     (GetMenu soundI text:)
  ```

  would return a pointer to the text corresponding to the soundI menu item.

08/11/88
- Install no longer passes the colon ":" to instgame.bat.

- New debugger command: 'S' shows the size and maximum and current stack usage of both the processor ('Proc') and PMachine ('PMach') stacks.

- New constants for show styles

  ```
  ;Picture change style constants
  (define  HWIPE       0)
  (define  HSHUTTER    0)
  (define  VSHUTTER    1)
  (define  WIPELEFT    2)
  (define  WIPERIGHT   3)
  (define  WIPEUP      4)
  (define  WIPEDOWN    5)
  (define  IRISIN      6)
  (define  IRISOUT     7)
  (define  DISSOLVE    8)
  (define  BLACKOUT    9)
  ```

  adding the "BLACKOUT" constant to any of the styles will blacken the screen in the converse manner before showing picture.

  ```
     (curRoom drawPic: PIC23 (+ IRISOUT BLACKOUT))
  ```

08/10/88

- Save/Restore game and sounds should now work much better together. Let me know if there are any restore game problems from now on (1 PM).

08/09/88

- PARSER, VC.EXE: Word derivation rules are now kept in a text file called DERIV.TXT, with a syntax similar to that of VOCAB.TXT. The vocabulary compiler has been modified so that it will compile both of these files when invoked. Comments are allowed in .TXT files following SCI syntax, ie. starting with a semicolon ";".

  A rule to handle plurals might be:

  ```
  (*men noun *man noun) ; ie. firemen/noun -> fireman/noun
  ```

  Multiple parts of speech can be specified for each pattern.There is a sample DERIV.TXT in the system directory.

  For the parser to function correctly, you must now have VOCAB files .000 (dictionary), .900(grammar) and .901(derivations) present at runtime.[Pablo]

8/08/88

- The restore: method of Game now takes an optional parameter. If this parameter is TRUE, the restore game dialog will be displayed only if there are saved games in the current save-game directory. This allows you to put up a restore dialog at the beginning of the game, which is a great convenience to the user. Just put the following in the init: method of your game:

  ```
     (if (not (GameIsRestoring))
        (theGame restore:TRUE)
     )
  ```

- The setCursor: and setSpeed: methods of Game now return the old values of the properties which they are replacing.

- New debug features!
    - Variables displayed in the debugger are now editable.
    - Breakpoint is back in an object-oriented reincarnation. Pressing 'b' in the debugger prompts for an object name (or ID), then for a method name. Once those are entered, execution begins and continues until execution enters the indicated method of the object, at which point the debugger is invoked. If no method name is entered (i.e. if you press Esc at the 'in method' prompt), a breakpoint is invoked whenever the indicated object becomes current.

8/05/88

- When adding a button to a Print, you MUST supply a value to return when that button is selected. This WILL make it possible for a TRUE/FALSE test instead of a switch.

  ```
  (if
     (Print   "May I have a K of code please?"
        button: {Be my guest} 1
        button: {Kiss my node pool!} 0
     )
     ; take a K of code and smile
  else
     ; sulk and be moody
  )
  ```

[Note that through no fault of either the compiler or the Print code, the above test will always test false.]

- Display default width is NOW, the required width of the message with NO word wrap. To enable word wrap, you must pass a WIDTH: argument that is greater than or equal to 0. If you pass 0, the width will be determined in the same ratio as the (Print) procedure.

- StatusLine should now be redisplayed on restart: and restore:.

- (GameIsRestarting) now returns RESTARTING/RESTORING if you're in the first animation cycle after a restart:/restore:.

- The wordFail: method of Game now gets two parameters: the word which was not recognized and the user input line.

```
(method (wordFail word input)
   ...
```

8/04/88

- Added command line switch (-r) to sciv. When used, will Alert user when a new volume is opened on the same physical media. Has no bearing on the file based version.

8/03/88

- Redefined "buttons" and their handling in Print procedure. You may add up to 5 buttons to a print message. They are added to the window, aligned horizontally, from left to right in the order that you specified them. If one of them is selected, Print returns the number of the button (1 through n) to you. You must take action accordingly as below:

```
(switch (Print "Please click on button A."
         button: {A} button: {B} button: {C})
   (0
      (Print "You pressed ESC")
   )
   (1
      (Print "Very good!")
   )
   (else
      (Print "Wrong button")
   )
)
```

  PLEASE NOTE: This change saved 142 bytes of memory.

- Refined modeless dialogs (Print dispose:). A "dispose:" print sets the global, modelessDialog to point to it. This variable, if not zero, WILL point to a dialog that can be disposed of via (modelessDialog dispose:) A timed dialog will also ensure that this global is zeroed when the timer times out. If a game is SAVEd, RESTOREd or RESTARTEd, SOMEBODY must dispose of this window or it will hang around forever OR in the case of save, not be valid when the game is restored.

8/2/88

- New save game interface, with more error checking.

- New kernel procedure (CoordPri yCoord). Returns the priority that corresponds to yCoord.

```
   ; make body the same priority as the feet in a split actor
   (body setPri: (CoordPri (feet y?)))
```

- Additional command line editing keys:

```
   CTRL-C clears entire line
   HOME   moves to beginning of line
   END    moves to end of line
```