The SCUMM Manual

abs

This function calculates the absolute value of the given parameter.

Syntax var=abs expression

Example xdist = abs(actor-x dr-fred - actor-x nurse-edna)

Also see

Things to Remember:

actor

This command is used to initialize and change the facets of an actor.
Multiple arguments can be combined into a single statement.

Syntax actor actor-name
[costume costume-name]
[default]
[talk-color color]
[step-dist x-coord,y-coord]
[text-offset x-offset, y-offset]
[elevation number]
[animation default]
[walk-animation choreography]
[stand-animation choreography]
[talk-animation choreography,choreography]
[init-animation choreography]
[animation-speed speed]
[scale number]
[zclip [to-nothing][to-zplane1][to-boxes]]
[ignore-boxes]
[follow-boxes]

CLICK HERE TO SEE MORE COMMANDS

[name string]
[width number][color color-slot is palette-slot]
[special-draw special-draw-select]
[stop]
[turn angle]
[face angle]

Example
```
actor bobbin  default costume b-swim-skin talk-color green
actor donovan color light-green is black sound splat
actor bobbin  elevation -10 walk-animation swimming \
        stand-animation treading talk-animation \
        start-talking, stop-talking \
        init-animation treading
actor indy    name "Indy" width 16 scale 50 step-dist 16,12
actor dr-fred ignore-boxes always-zclip 1 scale 255
actor indy text-offset -20,-80
```

Also see do-animation, put-actor, wait-for-actor, walk, stop-actor

Things to Remember:

• When initializing an actor, you should use at least costume, default and talk-color.

* Multiple arguments are executed first to last.

• Whenever you set ignore-boxes, it is advisable to also set the zclipping and the scale. The system does not know what to do with the actor, because it is not in a box. Therefore,the actor is going to randomly clip based on what it was doing last. Usually, when you say ignore-boxes you will say never-zclip scale 255 as well.

costume: Sets the current costume for the actor.

default: Sets the actor command back to the system defaults. This should be used every time a character is inited to prevent attributes from the last character that used that actor number from showing up.
These system defaults are:
talk-color white elevation 0 walk-animation 2 stand-animation 3 talk-animation 4, 5 init-animation 1 animation-speed 0 scale 255 step-dist 8,2 width 16 follow-boxes text-offset 0,-80

Actor class bits will be set to class bits of DEFAULT-ACTOR-CLASSES (class bits of actor #0)

talk-color: Talk-color is the color of the character's text on the screen.

step-dist: Step-dist takes two parameters, x and y, which tell how many pixels the actor should step per frame, when walking. The x controls walk steps left and right and y controls walk steps front and back. Step-dist is tied to the actor's scale.

text-offset:Each actor can have an offset to where dialog is displayed.

depth: Scripts can control actor display sorting order.

elevation: Actors automatically clip behind each other based on their vertical "y" position. The greater the “y” value the more ‘in-front’ an actor is. This command can be used, to change the elevation of an actor, without changing its "y" coordinate. The automatic clipping will still use the base "y" position for its calculations. This can also be used to raise or lower an actor without worrying about walk boxes.

animation default: Resets the init, walk, stand, start-talk and stop-talk anims back to the original settings.

walk-animation: The default for the walk-animation is chore 2. Using this command allows you to use animation from a different chore as the walk.

stand-animation: The default for the stand-animation is chore3. Using this command allows you to use animation from a different chore.

talk-animation: Talk-animation has two parameters. The first is for start-talking and the second is for stop-talking. The default for the talk-anim is chore4. When the talking stops, chore 5 is used. This is usually a closed mouth cell. Using the talk-animation command allows you to use animations from different chore cells.

init-animation: The default location for the init-animation is chore1. When an actor is put into a room using put-actor, the init-animation is activated. A costume change will also initiate init-animation. This command allows you to use animation from a different choreography instead of chore1.

animation-speed: Sets the number of frames for which each frame of the animation will actually be shown on the screen. It can only slow it down. 1 and 0 do nothing. 2 doubles each frame.

scale: The actor's scale can be set anywhere from 0 to 255. The 255 factor is the normal size of the actor. When scale is set to 0, the step-dist is automatically 0. The actor will not move but the system thinks it is trying to move. If invisibility is called for, use an "invisible" costume, not 0 scale. Although the actor's step-dist works in conjunction with scale, it may be appropriate to change the step-dist if you change the scale, because the scaling perspective sometimes looks off. Beware that the box scaling may interfere with the actor scaling.

zclip: There are now a series of defines inside scummsys.def including to-nothing, to-zplane1, to-boxes, and to-text-plane.

ignore-boxes: Setting an actor to ignore-boxes will cause the actor to ignore all boxes in the room. Instructing the actor to walk to an x,y coord will cause the actor to head directly to that point. Because the actor is ignoring all box information, it does not know how to scale or zclip. Therefore, zclipping and scaling info must also be provided.

follow-boxes: Setting an actor to follow-boxes will cause the actor to pay attention to the regular boxing information. It turns ignore-boxes off.

name: Sets the name seen on the screen, when the cursor touches the actor.

width: The actor's on-screen width in pixels. Defaults to 16. Very rarely will this be used for actors of normal proportion

color: This component of the command substitutes one color for another within the actor's costume.

special-draw: Give the capability to switch between different actor drawing

routines. Current options include SPECIAL-DRAW-normal, shadow, shadow-clip, and multi-shadow.

stop: Stops actor who was turning or walking. Issues a stand command

turn: Turns actor the specified angle 0-359 degrees

face: Snap the actor to face specified angle 0-359 degrees

walk-pause: Allow the scripts to pause an actor

walk-resume: Actor resumes walking to destination

volume: Controls the actor's voice volume 0-127

frequency: Controls the actor's voice frequency default 256

pan: Controls the actor's voice pan 0-127 center is 64

actor-box

This function returns the walk box of the specified actor.

Syntax var=actor-box actor-name

Example last-box = actor-box indy

break-until (actor-box selected-actor > 7)

if (actor-box selected-actor > 7) {

Also see actor, walk.

Things to Remember:

* This is not the same as in-box. Actor-box indicates the box that the actor is actually using -- not just the box that he is standing in. It is possible to have overlapping boxes. In which case the actor may be standing in several boxes at once; however, he is only actually using ONE of those boxes.

* If the actor is ignoring boxes, it will return 0.

actor-chore

This function returns the last choreography executed by the actor. Includes walks and talks.

Syntax var=actor-chore actor-name

Example if (actor-chore lechuck is-not chuck-reach-low {

Also see actor, do-animation.

Things to Remember:

actor-costume

This function returns the costume of the specified actor.

Syntax var=actor-costume actor-name

Example if (actor-costume lechuck is-not chuck-punching) {

if (actor-costume jojo is monkey-skin) {

Also see actor.

Things to Remember:

actor-depth

This function returns the depth of the specified actor.

Syntax var=actor-depth actor-name

Example depthVal = actor-depth seagull

foo = ((actor-depth special-effect) + 1)

Also see actor.

Things to Remember:

actor-elevation

This function returns the elevation of the specified actor.

Syntax var=actor-elevation actor-name

Example height = actor-elevation seagull

foo = ((actor-elevation special-effect) + 1)

Also see actor

Things to Remember:

actor-facing

This function returns the directional facing of the specified actor.
This value is an angle with NORTH at 0, EAST at 90 etc.

Syntax var=actor-facing actor-name

Example head-facing = actor-facing imaginary-actor3

if (actor-facing selected-actor is-not NORTH) {

if (actor-facing dock-keeper is WEST) {

Also see actor, walk

Things to Remember:

actor-inventory

This function returns the number of items in the specified actor's inventory as well as the items.

Syntax count=actor-inventory actor-name

item = actor-inventory actor-name in-slot slot

Example inv-size = actor-inventory selected-actor

obj = actor-inventory guybrush in-slot count

Also see

Things to Remember:

- Replaced inventory-size and find-inventory

actor-moving

This function returns whether the actor is stopped or moving.

Syntax var=actor-moving actor-name

Example foo = actor-moving felon1

if (!actor-moving cook) {

Also see actor, walk.

Things to Remember:

• The function returns one of the following two values:
moving = !0
stopped = 0

actor-room

This function returns the room number of the specified actor.

Syntax var=actor-room actor-name

Example if (actor-room cook is-not bar) {

if (actor-room pirate-num is the-void) {

leader-room = actor-room selected-actor

Also see actor, put-actor.

Things to Remember:

actor-scale

This function returns the scale of the specified actor.

Syntax var=actor-scale actor-name

Example if (actor-scale lechuck > 30) {

Also see actor.

Things to Remember:

actor-scale-array-string

System variable. Allows users to control the actor scale table.

Syntax

Example

Also see

Things to Remember:

actor-talking

System variable hold the last actor who spoke.

Syntax

Example

Also see

Things to Remember:

actor-width

This function returns the width of the the specified actor.

Syntax var=actor-width actor-name

Example dist = (actor-width jojo / 2)

give-prox = ((actor-width current-noun2 / 2) + 4)

Also see actor, proximity.

Things to Remember:

* Controls how close actors get when walking one actor to another.
* Default width is 24 pixels.

actor-x

This function returns the x room coord of the specified actor.

Syntax var=actor-x actor-name

Example if = (actor-x fishy > 30)

where-am-i = actor-x selected-actor

Also see actor.

Things to Remember:

* Returns room coordinates -- not screen coordinates.

actor-y

This function returns the y room coord of the specified actor.

Syntax var=actor-y actor-name

Example foo = ((actor-y selected-actor) + step-size)

Also see actor.

Things to Remember:

* Returns room coordinates -- not screen coordinates.

actor-zplane

This function returns the zplane of the specified actor.

Syntax var=actor-zplane actor-name

Example if (actor-zplane lechuck = 3) {

Also see actor.

Things to Remember:

actor.def

Assigns a number to each actor. Global chores are also defined here. Created by MAKEDEFS.

Syntax

Example

Also see

Things to Remember:

.anm

Files generated by Deluxe Paint Animator.

Syntax

Example

Also see

Things to Remember:

array

Arrays can be one or two dimensional. Arrays can be static or dynamic.

Syntax compile time (static) array:
variable variable-name[array-size]

run-time (dynamic) array:
dim array variable-name[array-size]

a list assignment:
variable-name[] = value,value, value, ...
or
string = "abcdefg"

a two dimensional array:
dim array variable-name[array-size][array-size]
variable-name[1][] = value, value, value, ...
variable-name[2][] = value, value, value, ...

Example variable actor-stuck[15]

dim array hit-points[100]

hit-points[] = 1,2,3,4,5,6,7,8,9

dim array maze-map [100][100]
maze-map[1][] = 1,2,3,4,5,6,7,8,9
maze-map[2][] = 1,1,1,1,1,1,1,1,1

Also see dim, undim, variable

Things to Remember:

- Arrays are just simple variables holding pointers. Therefore, they can be assigned to other variables, passed to scripts as parameters and assigned to local variables. The following are legal statements:

```
dim array foo[10]
bar = foo
bar[1] = 5
```

- Strings can also be used as arrays. Strings are really byte arrays terminated by a zero. They can be defined with a DIM statement or as part of an assignment. Characters in the string can be accessed as in an array.

array-shuffle

This command randomizes elements in a one dimensional array

Syntax array-shuffle foo[10] to foo[20]

Example

Also see

Things to Remember:

bit variable

A variable with a value of 1 or 0, true or false.

Syntax bit variable variable-name

Example bit-variable talked-about-leaving-flag

Also see

Things to Remember:

blast-object

A very fast drawing routine that can draw lots of things to the screen at any x,y SCREEN coordinate.

Syntax blast-object object at x-pos , y-pos

Example blast-object platter at 120, 130

Also see draw-object, state-of

Things to Remember:

- This isn't currently fully functional in the system.

But when they are:

- Blast-objects only stay on the screen for one frame.
- Their transparent color is 255.
- Blast-objects may cause some problem with the text plane.

blast-text

A very fast text drawing routine that can display text at any x,y position.

Syntax blast-text "string" at x-pos , y-pos [charset font color col wrap]

Example blast-text "hello world" at 120, 130

Also see print-text

Things to Remember:

- Blast-text only stay on the screen for one frame.
- wrap keeps the text within the screen boundaries

both-buttons-clicked-key

System variable. Key passed if both controller buttons pressed.

Syntax

Example

Also see

Things to Remember:

break-here

Simple break away command. Essential to our method of multitasking.
Can be used to time animation sequences.

Syntax break-here [number]

Example
```
break-here
break-here 4
break-here (k * 9)
```

Also see do-animation, sleep-for, break-until, do

Things to Remember:

• Breaks are frame driven. That makes them useful in timing animation. Sleep-for uses real time and is not recommended for animation, because of the wide range of computer power on our target machines.

• It is possible to condense the break-here's into one statement when they follow each other. This is accomplished by placing a number after the break-here.

break-until

A combination of a break-here and a do-until.

Syntax break-until (condition-met)

Example break-until (!script-running done-with-head)

Also see break-here, do[until] break-while

Things to Remember:

- The following two examples are identical.

```
do
   break-here
} until  (condition)

break-until (mess-flag)
```

- 1 break will ALWAYS occur

break-while

breaks will occur if condition met

Syntax break-while (condition-met)

Example break-while(!script-running done-with-head)

Also see break-here, do[until] break-until

Things to Remember:

• The following two examples are identical.

```
while (mess-flag) {
  break-here
}

break-while(mess-flag)
```

- The condition is checked BEFORE the break is executed.
- NO breaks will occur if condition is FALSE

build-sentence-script

System variable. Script to run when button hit.

Syntax

Example

Also see

Things to Remember:

camera

Stop the camera from moving

Syntax camera pause
camera resume

Example

Also see

Things to Remember:

camera-accel-x/y

System variables. How quickly the camera accelerates.

Syntax

Example

Also see

Things to Remember:

camera-at

This command sets the camera position in a room. The values indicates the midpoint of the screen.

Syntax camera-at x-coord [y-coord]

Example camera-at 320

camera-at 320, 200

Also see camera-follow, camera-pan-to, wait-for-camera

Things to Remember:

• 320 is the center of a 640 pixel wide screen so, camera-at 320 will position the camera at the far left side of a multi-screen room.

* This disables camera-follow.

camera-follow

This command tells the camera to follow the specific actor around the room.

Syntax camera-follow actor-name

Example camera-follow indy

Also see camera-at, camera-pan-to, wait-for-camera

Things to Remember:

• Once set, the camera will continue to follow the actor within that room. If this actor is not the selected-actor and goes into another room, you must change to that room, using either current-room or come-out-door. The camera will then continue to follow the actor in the new room.

•If the actor used as the parameter is in another room then the system will fade out of the current-room, and fade into the room in which the actor resides.

•Any other camera command will disable camera-follow.

• Camera-follow is important when a room is bigger than one screen or when there are multiple actors that can be used by the player and to ensure that the camera will follow the actor when they leave the room

camera-follow-actor

System variables which holds the current actor being followed.

Syntax

Example

Also see

Things to Remember:

camera-max-x/y

System variables which holds the maximum position for the camera.

Syntax

Example

Also see

Things to Remember:

camera-min-x/y

System variables which hold the minimum position for the camera.

Syntax

Example

Also see

Things to Remember:

camera-pan-start-x/ydist

System variables. How far from actor before pan starts.

Syntax

Example

Also see

Things to Remember:

camera-pan-to

This command tells the camera to smoothly pan to a new position in the room.

Syntax camera-pan-to x-coord [y-coord]

Example camera-pan-to 100
wait-for-camera

camera-pan-to 200, 400

Also see camera-follow, camera-at, wait-for-camera, camera-pan-speed

Things to Remember:

•The current position can be watched using the camera-x /y system variable.

• This statement can cause problems. The wait-for-camera statement should be used in conjunction with camera-pan-to, as the wait-for-camera statement waits until the camera system is stable. You don't want the code that follows to be run before the camera gets to its pan position.

• Make sure that when the player enters into a dialog in a scrolling room that you either use the say-line overhead statement or that the camera is repositioned so that the actors are more or less centered on the screen. This looks better and prevents ugly word wrapping. When the dialog is over, do a camera-follow selected-actor.

• Be aware of possible lockups due to a break-until camera-x = foo never becoming true. Make sure you are checking for a value that the camera-pan-to will hit. Camera-x increments by 8 so make sure you check a value which can be returned.

camera-script

System variable. Script to run when camera moves out of sync with other scripts.
Used for parallax effects.

Syntax

Example

Also see

Things to Remember:

camera-speed-x/y

System variables. Maximum speed for the camera.

Syntax

Example

Also see

Things to Remember:

camera-x

System variable which holds camera x position in pixels. 320 is center screen.

Syntax

Example

Also see

Things to Remember:

camera-y

System variable which holds camera y position in pixels. 200 is center screen.

Syntax

Example

Also see

Things to Remember:

case

Multiple choice branching statement.

Syntax

```
case case-name {
 of number {
  [statements]
 }
 of number {
  [statements]
 }
 default|otherwise {
  [statements]
 }
}
```

Example

```
case word-message {
  of 1 {
   say-line "In Olde English"
  }
  of 2 {
  say-line "In Latin"
  }
  default {
  say-line "Oh, I don't know."
  }
}
```

Also see if

Things to Remember:

- Case is converted to a series of if statements by Scumm.
- The case statement must have at least one "of" <value> statement.
- Default and otherwise are the same. Either one can be placed as the last choice in the case statement and will be used if all the other options are negative.
- If you are using a "compile if/endif" (example: #if MAC) statement within a case statement then it cannot be placed between the case and of portion of the code.

```
case (case-name) {
      of (# choice) {
      ; compile if (#if) goes here.
      ; compile endif (#endif) goes here
      }
}
```

chain-script

This command loads and executes a new script and stops the calling script.

Syntax chain-script [bak|rec] script-name [(] [^1[,]^2] [)]

Example chain-script ed-to-front-door

Also see stop-script, start-script

Things to Remember:

- Chain-scripts can be background or recursive.
- Chain-script stops the calling script.

charset*

The charset command sets the current character set being used. The system will load the charset onto the heap if it is not already there.
*UNDER RECONSTRUCTION

Syntax charset [charset-name]
[color transparent-color color color color]

Example charset fat-font

charset test-font color 0 35 126 light-gray

Also see nuke-charset, print-line, say-line, print-text, verb

Things to Remember:

* To remove a charset from the heap use nuke-charset.

* The colors are passed in the order that they appear in SPIT. The first color passed MUST be for transparent color. By convention, the second, third and fourth colors are the font color, drop shadow, and highlight respectively.

* It is possible to change charsets in the middle of a string of text with %f. For example: say-line “Hi, %ffat-font%how are you.”

chore

A choreography.

Syntax

Example

Also see

Things to Remember:

class-of

This command is used to set the classification of objects and actors.
It can also be used as a function to check the classification.

Syntax class-of object is object-class
class-of actor is actor-class

As a function:
if (class-of object is class) {
if (class-of actor is class) {

Example class-of head-bighead-door is UNTOUCHABLE LOCKED

class-of indy is WALK-FLIP-X

As a function:
if (class-of head-bighead-door is UNTOUCHABLE) {
    < do a bunch of stuff >
}

When initializing an object use 'class is':
object head-bighead-door
class is UNTOUCHABLE

foo = (class-of TOUCHABLE, GIVEABLE)

class-of chair UNSET-ALL touchable

Also see set-box , state-of

Things to Remember:

- Each object can be a member of any (or none) of the thirty-two possible classes, with class 0 being the default (not a member of any class except touchable).

- Object classes are set up within the object code with the expression class is. They are changed throughout the game with the class-of statement

- Class UNTOUCHABLE is the only class that the system knows about and must have to operate. Although this may change from project to project.

- Possible actor classes are UNTOUCHABLE, PLAYER-ONLY, WALK-FLIP-X and WALK-FLIP-Y.

- Classes are listed in CLASSES.DEF and may change from project to project.

- Several classes can be SET or CHECKED at the same time

- You can clear all class bits with a class-of foo UNSET-ALL.

- DEFAULT-ACTOR-CLASSES are the class bits for actor 0 and when you default an actor, it's class bits will be set to this.

#code on l off

Placing these in you code will cause the compiler to output source and object code to the screen. Used for debugging and the development of the language.

Syntax #code on | off

Example #code on
#code off

Also see #if #else #ifdef #endif #lex on l off

Things to Remember:

- This is useful for seeing what code the compiler is generating.

- To see the entire file dumped to the output, use the -C option with scumm
SCUMM -C inside.scu

come-out-door

This command is used to change rooms.
Places the selected-actor in a new room.

Syntax come-out-door object-name \
[walk x-coord,y-coord]

Example come-out-door hall-office-door

come-out-door hall-office-door walk 114,111

Also see entered-door, put-actor, update-inventory, enter code, exit code, selected-actor.

Things to Remember:

• entered-door will be set to the object that the actor enters through.

• The selected-actor will be placed at the object's use-position facing the opposite direction.

• Come-out-door is equivalent to put-actor at object-name [at x-coord, y-coord] except that the actor faces the opposite of the object's use position.

• Adding a walk statement to the same line is also possible. This is used to bring the actor further into the scene on a room transition.

• Come-out-door updates the inventory and runs the appropriate exit and enter code. It must be the last statement in an object or local-script -- any other code within the object or local script will not be executed.

• Do not use this in a local cut-scene. The main reason is that a come-out-door changes the current-room. This runs the exit code for the room you're in, and the enter code for the room you're entering. If you use a come-out-door within a local cut-scene, the script is killed when you exit the room, but the cut-scene status is unchanged. The cursor will still be set to soft-off, the userput is still soft-off and the cut-scene state is 1. The cut-scene will never end. This will generate an error message saying script stopped with active cut-scene. Be especially wary of of cutscenes written to accommodate complex entry or exit animations.

• CAUTION: Camera-follow may get clobbered by camera commands in the enter code.

#comment

Allows the script to display messages during compile time

Syntax #comment "string"

Example #comment "hey remember to clean up this routine"

Also see #if #else #ifdef #endif #lex on l off #error

Things to Remember:

- This also outputs the file and line number where the comment occurred

controller

System variable. Type of game controller.

Syntax

Example

Also see

Things to Remember:

.COS

Run length, compacted versions of .cst files. The system requires that animations be in .cos format.

Syntax

Example

Also see

Things to Remember:

costumes.def

Every costume must be defined in this file. You can force the system to create this file by running MAKEDEFS.

Syntax

Example

Also see

Things to Remember:

.cst

Files generated by CYST.

Syntax

Example

Also see

Things to Remember:

current-room

This command changes the camera to a new room. In the process it causes the exit code from the last current room to be run and then the enter code in the new current room.

Syntax current-room room-name

Example current-room driveway

Also see

Things to Remember:

- The new room is automatically loaded off the disk if it isn't already on the heap. The costume of any actor already placed in the room is also loaded in.

- NEVER use current-room in a local cut-scene. This command closes the room that you are presently in. All local scripts and code in objects are stopped.

current-

System variable for the user's computer time. Must call get-time-date before these are read.

Syntax

Example

Also see

Things to Remember:

cursor

This command sets the shape of the cursor.

Syntax cursor object-number [image flames-state]

Example cursor flames
cursor boss-cursor-shape
cursor flames image 5

Also see put-cursor

Things to Remember:

- Resets the hotspot and the transparencies.
- Object’s use position becomes the default cursor hot spot.

cursor hotspot

This command sets the hotspot of the cursor.

Syntax cursor hotspot x y

Example cursor hotspot 19, 12

Also see put-cursor

Things to Remember:

- Cursor hotspots are defaulted in Flem and can be set in flem.
- The x and y values are offsets from the upper left hand corner.

cursor onloff

This command allows you to remove and disable the cursor during cutscenes.

Syntax cursor on|off|soft-on|soft-off

Example
cursor on
cursor off
cursor soft-on
cursor soft-off

Also see put-cursor

Things to Remember:

- The soft-on/soft-off are for embedded cutscenes.

- When a cut-scene starts, the system decrements the cursor flag and when the cut-scene ends, the system increments the cursor flag. Any time that the cursor is a positive number the player can use the cursor. The number 0 or a negative number will deactivate the cursor.

- on = set to 1, off = set to 0, soft-on = increments, soft-off = decrement

- Soft-on and soft-off can be the source of subtle cursor bugs. Be very cautious when using this command. Always be sure that for every soft- off there is a matching soft-on.

cursor transparent

This command sets transparent colors in the cursor.
It can be called multiple times for multiple transparent colors.

Syntax cursor transparent color

Example cursor transparent 125

Also see

Things to Remember:

- This is mostly used to drop out background colors when using inventory objects as the cursor.

cursor-room-x

System variable for the x position of the cursor in the room.

Syntax

Example

Also see

Things to Remember:

cursor-room-y

System variable for the y position of the cursor in the room

Syntax

Example

Also see

Things to Remember:

cursor-screen-x

System variable. Gives the location of the cursor in SCREEN coords

Syntax

Example

Also see

Things to Remember:

cursor-screen-y

System variable. Gives the location of the cursor in SCREEN coords

Syntax

Example

Also see

Things to Remember:

cursor-state

System variable.

Syntax

Example

Also see

Things to Remember:

cut-scene

A cut-scene is a computer controlled sequence. During a cut-scene, the player has no control of the game. All other scripts, except BAK scripts, are temporarily halted during the cut-scene.

Syntax

```
cut-scene [type]{
    [statements]
}
```

Example

```
cut-scene quick-cut {
    do-animation selected-actor get-up
    break-here 20
    actor selected-actor costume guybrush-skin
}
```

Also see script, start-script, stop-script, chain-script, jump, override, come-out-door, current-room,

Things to Remember:

The 3 basic types of cut-scene are:

no-verbs
This removes the interface during the cut-scene. On exiting the cut-scene the camera is following the selected-actor.

quick-cut
This leaves the verbs on the screen but still turns off the cursor. There is no camera-follow selected-actor at the end.

no-verbs-no-follow
This component of cut-scene is the same as the no-verbs section with the exception that camera-follow selected-actor is not implemented.

These parameters (no-verbs, quick-cut, no-verbs-no-follow) are just flags passed to the enter/exit cut-scene scripts. Other types can be added as needed to the start-cut-scene and stop-cut-scene scripts in main-scr.scu

Do not use the statements come-out-door or current-room in a local cut-scene.

Parenthesis around the cut-scene identifiers are optional.

A cut-scene must always exit through its end (closing bracket.)

cut-scene1-script

System variable. Script to run on start of cut-scene.

Syntax

Example

Also see

Things to Remember:

cut-scene2-script

System variable. Script to run on end of cut-scene.

Syntax

Example

Also see

Things to Remember:

CYST

The animation tool used primarily for walk talk animations. It breaks an animation down into limbs that can then be manipulated independently.

Syntax

Example

Also see

Things to Remember:

debug

Used to start windex or execute windex commands from scripts.

Syntax debug debug-level [windex command]

Example debug 0
debug 1 "actors"

Also see

Things to Remember:

- When used with a windex command, this does not break into windex. It just sends the appropriate information to windex. In the example above, it would send the Actor information to windex.

- You can trace small sections of code.

debug 1 “trace”
code
code
code
code
debug 1 “run”

- The debug command is used in conjunction with the “set debug level” in windex. If you set the debug level to 3, then sputm will break into windex on a “debug0” a “debug 1” a “debug 2” or a “debug 3” command. All debug commands with a greater level will always cause a break into windex. This is so you can leave frequently used breakspots compiled in, yet ignore them or acknowledge them at run time.

define

Used to define any symbol.

Syntax define symbol
define symbol = expression

define-script =
define-sound =
define-costume =

Example define reach-high = 9

Also see

Things to Remember:

- Used extensively to define animation names.
- Always use define constants for clarity and maintenance.
- define-script/sound/costume is the same as #define but used for clarity.

define-local-script

Local scripts do not usually need to be defined. However, in special situations, they must be defined using define-local-script. This allows you to mix auto-defining-scripts with predefined scripts.

Syntax define-local-script local-script-name

Example define-local-script init-congress

Also see

Things to Remember:

- Local scripts are self defining. However, if a reference is made to a local script higher up in the .scu, it then must be defined using define-local-script. This can happen if two local scripts call each other or if a global script calls a local script.

- If desired, all local scripts could be defined. But this is not required.

define-script

Used to define scripts.

Syntax define-script script-name

Example define-script follow-fred

Also see

Things to Remember:

delcos.bat

Created by makedefs. It contains all unused .cos files in your project's directory tree. Running this batch file will delete all these unused files. USE WITH CAUTION.

Syntax

Example

Also see

Things to Remember:

dellbm.bat

Created by makedefs. It contains all unused .lbm files in your project's directory tree. Running this batch file will delete all these unused files. USE WITH CAUTION.

Syntax

Example

Also see

Things to Remember:

delscu.bat

Created by makedefs. It contains all unused .scu files in your project's directory tree. Running this batch file will delete all these unused files. USE WITH CAUTION.

Syntax

Example

Also see

Things to Remember:

Deluxe Paint

The art program used for background art. Made by Electronic Arts.

Syntax

Example

Also see

Things to Remember:

Deluxe Paint Animator

The animation program used for 2D character animation. Made by Electronic Arts.

Syntax

Example

Also see

Things to Remember:

dependent-on (retired)

This command makes the state of one object dependent on the state of another object.

Syntax dependent-on object-name being object-state

Example dependent-on refrigerator-door being OPEN

Also see

Things to Remember:

- For example, a can of Pepsi inside a refrigerator. If the door is closed, the Pepsi would not be visible or touchable. The Pepsi is dependent on the refrigerator door being open.
- Dependencies are ignored by draw-object and blast-object.
- This might not work well with multiple state objects where the “HERE” state is to the right of the imagemin.

dim

This command dynamically allocates an array and stores a pointer to the array.

Syntax dim [local] [string]array variable [size]

two dimensional arrays:
dim array variable[size-of-array1][size-of-array2]
variable [1][] = value, value, value, ...
variable [2][] = value, value, value, ...

Example dim array hit-points(100)

dim local string array franklin-dialog-flags[15]

dim array maze-map[100][100]
maze-map[1][] = 1,2,3,4,5,6,7,8
maze-map[2][] = 1,1,1,1,1,1,1,1

Also see array, undim, variable

Things to Remember:

- Because arrays are declared at run-time, it is possible to reuse array variables.

- Because the DIM statement assumes that any value in hit-points (the example above) is an array pointer, it can be dangerous to do the following:

```
hit-points = 1000
dim array hit-points(100)
```

The DIM command will try and deallocate an array whose pointer is 1000 -- which doesn't exist.

- Because arrays are just simple variables holding pointers, arrays can be assigned to other variables, passed to scripts as parameters and assigned to local variables. The following are legal statements:

```
dim array foo(10)
bar = foo
bar[1] - 5
```

- Use UNDIM before the script ends when using local variables as arrays or the the pointer will be lost and the space for the array can never be recovered from the heap.

- Two dimensional arrays assigned lists of numbers must be declared with the DIM statement.

- local arrays are deleted when the script that created them ends.

display-version-key

System variable. Which key causes version displays to appear.

Syntax

Example

Also see

Things to Remember:

DK

The very powerful art tool for color reduction and manipulation. Most of the manipulation of palettes in finished art work is done in DK.

Syntax

Example

Also see

Things to Remember:

do [until]

The do construct is used to repeat a section of code. The do until command is used to repeat a section of code until a condition is satisfied.

Syntax

```
do {
 [statements]
}

do {
 [statements]
} until (condition-met)
```

Example

```
do {
    draw-object bozo image 1
    break-here
    draw-object bozo image 2
    break-here
}

do {
 [statements]
}until (proximity indy henry <= 26)

do {
   [statements]
} until (number in [1 2 3 4 5 6 ]
```

Also see for

Things to Remember:

• It will repeat indefinitely until the script it is in is stopped, either by a stop-script, a jump or, if it is in a local script, by a room transition.

• The do loop will be executed at least once before checking to if the until condition is true. To check the condition prior to executing the loop, use 'while.'

• It is possible to use set comparison in a do until loop.

• There should be a break somewhere inside the bracketed code or the computer will lock up

This is correct:

```
do {
     draw-object lights
     sleep-for delay jiffies
     state-of lights is R-GONE
     sleep-for delay jiffies
}
```

This would cause a lockup:

```
do {
     draw-object lights
     state-of lights is R-GONE
}
```

• Do until loops can be embedded within other do until loops or within a do loop.

```
do {
            do {
            [statements]
    } until
} until

do {
     do {
            [statements]
```

```
    } until
}
```

do-animation

This command tells an actor's costume which animation sequence to perform.

Syntax do-animation actor-name choreography

Example do-animation bernard ring-bell
break-here 4
do-animation bernard reach-low

Also see break-here, do-animation face-towards

Things to Remember:

- If animation sequences are strung together they must, usually, be broken up so that each frame is seen by the user. The example shows the break-here command as an effective tool for this purpose. Break-here is based on frames.

- The command sleep-for will also cause a pause in action; however, this uses "real" time instead or computer time. This means that on a slow computer it is possible for the animation to take longer than is allowed in the sleep-for command. Using sleep-for is not recommended for timing animations.

- To make code more readable, always use the defined choreography names instead of actual chore numbers.

do-animation face-towards

This command causes an actor to face towards another actor.

Syntax do-animation actor-name face-towards actor-name

Example do-animation hermit face-towards guybrush

Also see do-animation, wait-for-actor, break-here

Things to Remember:

- The actor will go through all the intermediate facings on the way.

- Putting do-animation face-towards into a loop will allow an actor to continue watching someone as they move across the screen.

Example:

```
do {
    do-animation max face-towards sam
     break-here
}
```

do-sentence

This command lets the programmer artificially construct a sentence which will execute as if the player had controlled it.

Syntax do-sentence verb-name object-name
do-sentence verb-name object-name with object2

Example do-sentence open alarm-exit
do-sentence pick-up envelope
do-sentence use video-game1 with quarter

Also see stop-sentence

Things to Remember:

- Do-sentence is most useful when a command must execute code already defined in another command.

- Do-sentence can also be used with current-noun1 and current-noun2 to save having to carry around a lot of code that's attached to a specific object.

- It's possible to string a series of do-sentences together. Each one is placed on the stack, and then they are all executed, one-by-one, in reverse order. The system then calls the current sentence-script repeatedly until the stack is gone.

- Note that the use of a preposition in do-sentence is an optional defined value which Scumm throws away. It is included for increased readability. Other prepositions which are defined are in and on.

Dpaint

see deluxe paint.

Syntax

Example

Also see

Things to Remember:

draw-box

This command paints a solid colored box on the screen.

Syntax draw-box x-coor,y-coor to x-coor,y-coor color color-name

Example draw-box 0,155 to 320,185 color black

Also see

Things to Remember:

• This can be used for bar graphics such as the fighting levels in Indy 3, or the lines on the travel maps, made by drawing a series of small boxes in a line.

• The first x, y coord is the upper right-hand corner of the box and the second (x), (y) coord is the lower left-hand corner.

• There is a problem with the draw-box statement during saving and loading. If a game is saved after a box is drawn, when it is reloaded, the drawn box will not appear on the screen since the state of the screen is not saved to disk. When drawing a line to the screen, it is a good idea to disable save-load.

• The box is drawn for one frame.

draw-object

This command draws an object to the screen.

Syntax draw-object object-name [at x-coord,y-coord]
[image state-number]

Example draw-object hull-flame1

draw-object hell-obj at 100,0

draw-object ghost-hand image 1

Also see

Things to Remember:

• Draw-object allows us to incorporate screen animations using multiple objects. The object can either be drawn at its original position (defined in flem), or a new x,y location can be specified. Note that the object's state 0, the one drawn into the background, can not be used in animation cycles.

• Draw-object just blasts the image onto the screen background. It does not matter what was there before. Animation sequences that are constantly moving are ideal for draw-object and not state-of. State-of is a very sophisticated object image manipulator which checks all the other objects to see if it is clobbering anything else. This takes time. Therefore, it is faster to use draw objects when nothing bad can happen.

• This command is also useful for making up new versions of rooms which have different features (pseudo rooms). For example, the same door object could appear in a different place on the screen in each version of the room. You would then have to position these objects every time you enter the room (using the enter code) or they will revert back to their originally defined location.

#else

A compile else follows a #if. It is checked at compile time.

Syntax #else
[stuff]

Example
```
#if COPY-PROTECTION
    override about-to-start-cp
#else
    < a whole bunch of statements >
#endif
```

Also see #if #ifdef #endif #code on l off #lex on l off

Things to Remember:

enter-room1-script

System variable. Script to run BEFORE enter code.

Syntax

Example

Also see

Things to Remember:

enter-room2-script

System variable. Script to run AFTER enter code.

Syntax

Example

Also see

Things to Remember:

entered-door

System variable indicate object that actor came out of.

Syntax

Example

Also see

Things to Remember:

environment variable

Set in scumm.ini file, these variables set the paths for the project. For example: ARTPATH=c:\indy5\ART\ causes all utilities using ARTPATH to search in this directory. Typing ART at the dos prompt will take the scriptor to the ART directory.

Syntax

Example

Also see

Things to Remember:

#error

Allows the script to display messages during compile time
Stops compilation

Syntax #error "string"

Example #error "hey remember to clean up this routine"

Also see #if #else #ifdef #endif #lex on l off #comment

Things to Remember:

- This also outputs the file and line number where the error occured.

exit-room1-script

System variable. Script to run BEFORE exit code.

Syntax

Example

Also see

Things to Remember:

exit-room2-script

System variable. Script to run AFTER exit code.

Syntax

Example

Also see

Things to Remember:

facing*

This function returns the actor or object's current facing angle.
Angles are 0-360 degrees with 0 being North, 90 being East

Syntax var=facing actor-name
var=facing object-name

Example head-angle = facing imaginary-actor3

if (facing selected-actor is-not 90) {

if (facing dock-keeper is NORTH) {

Also see

Things to Remember:

FACING was added in order to switch from Left/right/front/back to angles.
Now that angles are standard, the older actor-facing and object-facing
have been reintroduced.

fade-delay

System variable. Number of piffies (1/240 th of a second)between fade steps.
Default is 3.

Syntax

Example

Also see

Things to Remember:

fades*

This command controls the screen transition during the next room transition.
*nonfunctional in 640x480 games

Syntax fades fade-name

Example fades pixel-down

Also see

Things to Remember:

- Fades does not run a fade, it just sets the type of fade to be used next time a fade is run.

- The basic fades are none, iris, pixel and snap, which have both an up and a down version. These are used in combination to generate the fades that the scumm programmer uses.

- Types of fades can be added at anytime so you should check your header.def for a complete list of all available fades.

- Possible fades include the following:

cross-pixel (none-down + pixel-up)
This fade allows you to pixel-fade directly into a new room.

cross-iris (none-down + iris-up)
This fade irises up the new room on top of the old room

cut-to (none-down + snap-up)
This fade jumps straight to the new room with no black inbetween.

iris-to-black (iris-down + iris-up)
This fade irises down in the old room and irises up in the new room.

cut-to-black (snap-down + snap-up)
This fade jumps to black and then jumps to the new room.

pixel-to-black (pixel-down + pixel-up)
This fade pixels down to black and then pixels up in the new room.

find-actor

This function finds the front-most actor at the given SCREEN coordinates.

Syntax var=find-actor x-coord, y-coord

Example to-whom = find-actor cursor-screen-x,cursor-screen-y

Also see new command for pixel accurate find-actor

Things to Remember:

- If no actor is present at the x and y position, the function returns a 0.
- find-actor uses the dirty-rectangle for an actor. If you need greater precision, see Aric about the new command.

find-all-objects

This function returns an array listing all of the objects in the current room.

Syntax foo = find-all-objects room-name

Example

Also see

Things to Remember:

find-inventory*

This function returns the inventory object in the given slot of the specified actor.

Syntax var=find-inventory actor-name, slot

Example obj = find-inventory indy,sslot

Also see inventory-size, actor-inventory

Things to Remember:

- Slots are numbered starting at 0.

* Replaced with actor-inventory [act] in-slot [slot]

find-object

This function returns the front-most object at the given ROOM coordinates. If no object is present at the x and y position, the value returned is a 0.

Syntax var=find-object x-coord, y-coord

Example k = find-object 12, 140

Also see

Things to Remember:

- It ignores untouchable objects.

find-verb

This function returns the verb at the given SCREEN coordinates.

Syntax var = find-verb xcoord ycoord

Example var = find-verb cursor-screen-x cursor-screen-y

Also see find-actor, find-object

Things to Remember:

- Make sure you use cursor-screen-x and not cursor-room-x.

Flem

The room layout tool for objects and walk boxes.

Syntax

Example

Also see

Things to Remember:

for

The variable is initialized with the first value. The system continues through the loop and on each pass, the variable is incremented (++) or decremented (--) until it equals the last value. It is possible to have for loops with sets.

Syntax

```
for variable = value to value ++ | -- {
 [statements]
}

for variable = [expression...] {
  [statements]
}
```

Example

```
for k = dialog-1 to dialog-9 ++ {
 verb k off
}

for i = 255 to 85 -- {
    actor old-purple-tentacle scale i
    break-here
}

for i = [first-door middle-door last-door] {
    state-of i is OPEN
}
```

Also see do

Things to Remember:

• There must be a space between the step increment (++, --) and the following brace.

This is correct:
for k = 0 to 1 ++ {

This is incorrect:
for k = 0 to 1 ++{

• Nested for loops are permitted in Scumm.

• If a script that is doing a for list loop is killed or ends before the loop is done, a small array will be left on the heap. To prevent this, do not use breaks or sleeps in a for list loop. If you are using breaks in for list loops, watch out for “out of array pointers” scumm error.

frame-jiffies

System variable. Real jiffies that the last frame took to me displayed.

Syntax

Example

Also see

Things to Remember:

freeze-scripts

This command is called by the cut-scene statement and freezes all scripts except BAK scripts. This command is used in the system level scripts, cutscenes,enter and exit scripts, dialogs.

Syntax freeze-script

freeze-script freeze-all-but-me

Example freeze-scripts

freeze-scripts freeze-all-but-me

Also see unfreeze-scripts, start-script, cut-scene, enter code, exit code, dialogs

Things to Remember:

- In normal scripting this command is rarely used or needed.
- If you must use it, don't forget to eventually issue an unfreeze-scripts command.
- freeze-all-but-me does what it implies.

game-loaded

System variable set if game loaded from the heap.

Syntax if (game-loaded)

Example
```
save-game
break-here
if (game-loaded == FALSE) {
   new-current-room some-closeup
}
```

Also see save-game, load-game.

Things to Remember:

get-time-date

This command causes the current-time variables to be set.

Syntax get-time-date

Example

Also see current-year, current-month, current-day, current hours, etc.

Things to Remember:

#if

A compile if is checked at compile time.

Syntax #if expression
#endif

#if “string” is “string”
[stuff]
#else
[stuff]
#endif

Example #if INTERNATIONAL_VERSION
#if CD
#if COPY-PROTECTION

Also see #ifdef #else #endif #code on l off #lex on l off

Things to Remember:

- It is also possible to do nested compile if's.
- It always needs a corresponding #endif.
- Macros are evaluated before #if. So do not use macros inside of a #if statement.

if [else]

If the condition presented is met, the code within the construct is executed. It the condition is not met, the code is skipped.

Syntax

```
if (condition met) {
 < statements >
}

if (condition met) < single statement >

if (condition met) {
 [statements]
} else {
 [statements]
}
```

Example

```
if (sheriff-met) {
 statements
}

if (xpos > 300) xpos = 300

if (not script-running good-shot and sank-my-own-ship) {
 start-script cliff-hermit-dialog
} else {
 start-script cliff-crew-dialog
}

if (selected-room in[basement den kitchen]) {
   statements
}
```

Also see

Things to Remember:

- Statements in parentheses are evaluated to true/false condition.
- Complex conditions can be constructed using or and and.
- The condition within the parenthesis can be a single parameter or more involved conditions.
- The if condition can take many different forms:

```
if (not second-time-on-melee) {
if (noun2 is root-beer) {
if (where-am-i < 320) {
if (actor-x cook > 310) {
if (button == 2) {
if (x < 1 or x > 10) {
if ((time-to-go is 5) and (script-running door-bell-ringing)) {
```

- Nested if statements are allowable.

```
if (item >= '0') {
      if (item <= '9') {
            return-value is (item - '0')
      }
}
```

- It is possible to use set comparison in an if statement.

if in

This function checks to see if an element is in a set.

Syntax if (expression in [expression [,] ...]) {

Example
```
if (current-room in [den, kitchen, hallway, laundry] {
    < do stuff >
}
```

Also see sets

Things to Remember:

#ifdef

Conditionally compiles code if the symbol is defined.

Syntax #ifdef symbol

Example
```
define french = 1
#ifdef french
```

Also see #if #else #endif #code on l off #lex on l off

Things to Remember:

.ifo

Files generated by Flem which contain all room information such as boxes, scaling, spots, objects, room size, number of zplanes, etc.

Syntax

Example

Also see

Things to Remember:

in-box

This function returns whether or not the point at which the actor is standing is located inside the specified box.

Syntax if (actor in-box box) {

Example
```
if (selected-actor in-box 5) {
    < do a bunch of stuff >
}
```

Also see

Things to Remember:

- This must be used in conjunction with an if statement.

- This is NOT the same at actor-box. Actor box refers to only the ONE box which the actor is using to get scaling, connectivity, etc. information. In-box refers to ALL boxes which contain the point at which the actor is standing.

include

Includes other files within the file. This makes it possible to break one file into several. It also makes it possible to include information which must be in all files such as that in header.def

Syntax include “filename”

Example include "header.def"
include "samdial.scu"
include "%BINPATH%scummsys.def"

Also see

Things to Remember:

• Header.def must be included in all .scu files. Header.def in turn includes a number of other files.

• Include is often used to separate long scripts out of an .scu and put them into their own file. Dialogs are often included in this way.

• Includes must be inserted before the info that they contain is referenced.

• Makemake only checks a single level of includes, so if “office.scu” include “samdial.scu” and “samdial.scu” in turn includes “bulbdial.scu”, then changes to “samdial.scu” will cause recompile of “office.scu” but changes to “bulbdial.scu” will not.

• %PATH% can be used with any environment variable to include language defines or other files based on path selection.

inventory-size*

This function returns the number of items in the specified actor's inventory.

Syntax var=inventory-size actor-name

Example inv-size = inventory-size selected-actor

Also see actor-inventory

Things to Remember:

* Replaced with actor-inventory [act].

jiffy

1 jiffy = 1/60 of 1 seconds

Syntax

Example

Also see

Things to Remember:

jiffy1

System variable incremented one a frame with jiffy count.

Syntax

Example

Also see

Things to Remember:

jiffy2

System variable incremented one a frame with jiffy count.

Syntax

Example

Also see

Things to Remember:

jiffy3

System variable incremented one a frame with jiffy count.

Syntax

Example

Also see

Things to Remember:

joystick-x/y

System variable. If controller is a joystick, the current x/y position.

Syntax

Example

Also see

Things to Remember:

jump

Used to jump from one section of code to another within a script or an object's code.

Syntax jump label-name

Example

```
    jump wait-for-awhile
    [statements]
wait-for-awhile:
    [statements]
    if (script-running kitchen-door-clicked) {
        jump wait-for-awhile
    }
    [statements]
```

Also see cut-scene, script, dialog

Things to Remember:

- A label must have a colon after it, but don't include the colon in the actual jump statement.
- Jump is used extensively in dialogs.
- Never jump out of a cut-scene where you bypass the cut-scene close brace. This is because a cut-scene launches a number of scripts that control the cursor and userput and a cut-scene must exit properly for the cursor and userput to be reset correctly.
- Scumm convention is for label names to be placed flush left.
- Labels are a value with indicates how far they are from the beginning of the script.
- Labels are local to a script. However, they remain in memory as long as the room is in memory. Therefore each must have a unique name in that room.
- If you attempt to jump to a label that is in another script, the system will read the value stored in the label and jump you to the point indicated in the script you are current in. This can cause all sorts of problems that may or may not be immediately detectable. Make sure that all labels have unique names and that all jumps are to the correct labels.

K-of-heap

System variable. Size of sputm heap in K.

Syntax

Example

Also see

Things to Remember:

kludge

Commands in training. These are run as macros. See kludge.def for current macro commands.

Syntax kludge kludge-name

Example see kludge.def... never hard-code numbers since they may change!

Also see

Things to Remember:

- These are set up so that commands can be added and tested without changing the compiler. If the commands are found to be useful, they will eventually be added as real commands to the system.

- The kludge command should NEVER be used directly by scripters. A kludge.def file is now included with the system that keeps the kludge commands properly named and this should allow much greater flexibility.

- A listing of the available kludge commands, look at kludge.def in your scummbin.

.lbm

Background art files.

Syntax

Example

Also see

Things to Remember:

left-button-state

System variable. Scripts can get button position. 1 is down, 0 is up.

Syntax

Example

Also see

Things to Remember:

#lex on l off

Used for debugging. It takes text and converts it to tokens.

Syntax #lex on | off

Example #lex on
#lex off

Also see #if #else #ifdef #endif #code on l off

Things to Remember:

.lfl

Contain compiled code for each room.

Syntax

Example

Also see

Things to Remember:

lflfile

Created by MAKEMAKE. It is used to show dependencies used by MAKE to determine if a room must be recompiled.

Syntax

Example

Also see

Things to Remember:

load-

This command loads the specified item from a disk onto the heap.

Syntax

```
load-costume  costume-name
load-room     room-name
load-script   global-script-name
load-sound    midi-effect
load-charset  charset-name
```

Example

```
load-costume  henry-dead-skin
load-room  zep-hall
load-script  butler-dialog
load-sound  crash-sound
load-charset  fat-font
```

Also see

Things to Remember:

load-game

This command loads the saved game from the heap

Syntax load-game
load-game no sounds

Example load-game

load-game no sounds

Also see save-game, game-reloaded

Things to Remember:

• Music is ALWAYS saved, but the scripts can load a game without changing the music stream. This allows saves to be used more seamlessly.

• Someday slot might be added.

• game-reloaded allows scripts to branch differently based on whether a game was loaded.

load-lock-

The load-lock macros loads items onto the heap and then locks them.
Macros defined in scummmac.def.

Syntax load-lock-costume costume-name
load-lock-room room-name
load-lock-script global-script-name
load-lock-sound midi-score

Example load-lock-costume sheriff-skin
load-lock-sound door-open
load-lock-room cu-brush
load-lock-script see-parrot

Also see

Things to Remember:

• This command emits a load-item followed by a lock-item. This may be replaced by a macro instead.

• Even loading and locking an item does not guarantee that it will stay on the heap, but it will only be removed after all unlocked items are removed.

• Remember to unlock items when you don't need them.

• Because of time stamping, this usually is no longer necessary. Use only for frequently loaded items to prevent CD skipping.

local variable

A variable that is local to a script.

Syntax local variable variable-name

Example local-variable k i counter

Also see

Things to Remember:

lock-

This command sets a flag which prevents an item from being removed from the heap once it's loaded in.

Syntax
lock-costume costume-name
lock-room room-name
lock-script global-script-name
lock-sound midi-file

Example
lock-costume hole-indy-skin
lock-room beach
lock-script edna-chases-kid
lock-sound kid-screams

Also see

Things to Remember:

- Make sure you unlock the item when it doesn't need to be in memory any longer.

- It is still possible to have a locked item thrown off the heap if the heap is full of locked items. This will generate a run-time warning if Windex is running. However locking it means that all unlocked items will be thrown off first. So remember to unlock items when they are no longer needed.

- If something is already loaded, it will not be loaded again. If something is locked but not loaded (yes, it is possible) it will become locked on the heap, as soon as it is loaded.

- Because of time stamping, this usually is no longer necessary. Use only for frequently loaded items to prevent CD skipping.

macro

Allows scripters to create more complex commands

Syntax macro macro-name ^1 ^2 {
  command ^1 ^2
}

Example macro wait-for-sound ^1 break-until (!sound-running ^1)

Also see

Things to Remember:

macros.def

Frequently needed functions that have not yet been implemented into the system are written as macros in this file.

Syntax

Example

Also see

Things to Remember:

make.exe

A DOS utility which is called by mk and mkall.

Syntax

Example

Also see

Things to Remember:

makedefs.exe

Makes the objects.def, scripts.def, costumes.def and sounds.def files from the .scu files

Syntax

Example

Also see

Things to Remember:

makemake.exe

Makes files (roofile and lflfile) that MK.BAT and MKALL.BAT use. It also builds the dependencies for .cos, .sou and INCLUDE files. It must be run every time a new room is added.

Syntax

Example

Also see

Things to Remember:

max-number-objects

System variable. Highest object # in the game.

Syntax

Example

Also see

Things to Remember:

memory-speed

System variable. Speed of memory copy.

Syntax

Example

Also see

Things to Remember:

message-going

System variable. 1 = message going.

Syntax

Example

Also see

Things to Remember:

min-jiffies

System variable governor for minimum number of jiffies per frame.

Syntax

Example

Also see

Things to Remember:

min-jiffies-per-char

System variable. Determines text delay on messages.

Syntax

Example

Also see

Things to Remember:

min-jiffies-per-message

System variable. Determines text delay on messages.

Syntax

Example

Also see

Things to Remember:

mk.bat

The batch file that brings all files up to date, checks to see which files have been changed and calls MAKE. Use mk when changes are made to the .scu only.

Syntax

Example

Also see

Things to Remember:

mkall.bat

The batch file that brings all files up to date, checks to see which files have been changed and calls MAKE. Use mkall when changes have been made to anything beyond the .scu file.

Syntax

Example

Also see

Things to Remember:

mmucus.exe

The room compressor for SCUMM. It builds .roo files from .lbm and .ifo data.

Syntax

Example

Also see

Things to Remember:

name is

This command is used at the beginning of every object definition to provide the name displayed when "%o" is used in a print command.

Syntax name is "object-name"

Example name is "cooking pot"

Also see

Things to Remember:

- If the object is never to be touched, the name string should be null.

- This should NOT be used is the object intended to be used as a cursor image object. Image and name are stored in the same slot so an object must have either and image or a name. Icon is is o.k. Because it associate the image of another object with the object rather than assigning both an image and a name to a single object.

new-name-of

This command changes the “string” name of an object.

Syntax new-name-of object-name is "string"

Example new-name-of cooking-pot is "empty pot"

Also see

Things to Remember:

• The "string" name is only the name the player sees on the screen. When you refer to the object in the program, use the unquoted object name (object-name).

• If an object has an image associated with it, using “image,” new-name-of will remove the association of the image with the object.

next-room

System variable. Only valid during EXIT code.

Syntax

Example

Also see

Things to Remember:

nuke -

This command sets the time stamp to be the oldest possible item on the heap. This will cause it to be thrown away first.

Syntax nuke-costume costume-name
nuke-room room-name
nuke-script script-name
nuke-sound sound-name
nuke-charset charset-name

Example nuke-costume purple-tentacle-skin
nuke-room hallway
nuke-script clock-running
nuke-sound kapoow
nuke-charset fat-font

Also see

Things to Remember:

- It sets the item's time stamp to the oldest item making it the next candidate to be removed if memory is needed.

object-facing

This function returns the use-angle (0-359) of the specified object.

Syntax var = object-facing object-name

Example no real examples since NO game has ever used this command.

Also see

Things to Remember:

object-image-height

This function returns the height in pixels of the object's image.

Syntax var=object-image-height object-name

Example var=object-image-height manuscript

Also see object-image-x, object-image-y, object-image-width

Things to Remember:

object-image-width

This function returns the width in pixels of the object's image.

Syntax var=object-image-width object-name

Example var=object-image-width manuscript

Also see object-image-x, object-image-y, object-image-height

Things to Remember:

object-image-x

This function returns the x coordinate of the object's image in ROOM coords

Syntax var=object-image-x object-name

Example var=object-image-x manuscript

Also see object-image-y, object-image-width, object-image-height

Things to Remember:

object-image-y

This function returns the y coordinate of the object's image in ROOM coords

Syntax var=object-image-y object-name

Example var=object-image-y manuscript

Also see object-image-x, object-image-width, object-image-height

Things to Remember:

object-running

This function is used to tell if the specified object is currently running.

Syntax var = object-running object-name

Example
```
if (not object-running practice-boxing) {
 start-script bak boxer-in-ring
}
if (object-running spinning-fan) {
 load-lock-script meeting-of-the-masters
}
```

Also see start-script, stop-script, freeze-script, unfreeze-script

Things to Remember:

object-x

This function returns the value of the x-coordinate of the use-position of an object in ROOM coords.

Syntax var=object-x object-name

Example dest-x = object-x obj

Also see object-y, object-image...

Things to Remember:

object-y

This function returns the value of the y-coordinate of the use-position of an object in ROOM coords.

Syntax var=object-y object-name

Example var=object-y manuscript

Also see object-x, object-image...

Things to Remember:

objects.def

A list and definition of all the objects in the game. It is created by MAKEDEFS.
The scriptor should never need to edit this file.

Syntax

Example

Also see

Things to Remember:

override

Used to jump over long sections of a game, especially cutscenes when a special key (usually escape) is pressed by the player. This lets the player skip long dialogs or long non-interactive sections of the game.

Syntax override label-name
override off

Example override skip-ghost-story

Also see jump, cut-scene, script, if

Things to Remember:

• The override-hit system variable can be checked before exiting a cut-scene to see if override was used to get to that spot. It will be true if an override was hit.
Example:

```
cut-scene           {
     current-room opening
     override skip-opening
     put-actor indy at 120,35 in-room opening
     [statements]
     current-room library
     put-actor indy at 35,10 in-room library
     [statements]

     if (override-hit) {
          current-room library
          put-actor indy at 35,10 in-room library
     }
     override off
}
```

• If the override key is hit the override is turned off, and the override-hit flag is set. If you reach the override label without the key being hit the override is still on. This can be dangerous! If the player hits override now, sputm will jump back up to the label. Overrides are cleared when the cut-scene in which the override is located ends, and with override off. Therefore don't put any commands that may cause a break after the override label but before you either end the cut-scene or say override off.

• Note that if an override is used in a cut-scene its label must be located within the same cut-scene. Don't use the override command to jump out of a cut-scene, bypassing its closing brace.

• When the escape is hit the override will immediately take over and jump to the label. This is dangerous because the escape key may be hit at any time after the override command. It is possible that scripts, sounds and animations will have started that need to be terminated. Make sure that all conditions that get set in the cut-scene are properly reset after the override is hit. Use an if (override-hit) to check and turn off as appropriate.

```
if (override-hit) {
     print-line " " ; erases text
     actor selected-actor costume guybrush-skin
     do-animation selected-actor stand
}
override off
```

• It is possible to have several overrides in a cut-scene. Only the last one executed will be used.

• Override off clears the override set. It also clears “override-hit.” So if it needs to be checked, that do it first.

override-hit

System variable. True if override hit.

Syntax

Example

Also see

Things to Remember:

override-key

System variable hold the key to be used for override.

Syntax

Example

Also see

Things to Remember:

owner-of

This command is used to assign ownership of an object to an actor. Can also be a function which returns a value of who is the owner of an object.

Syntax owner-of object-name is actor-name

As a function:
var = owner-of object-name
if (owner-of object-name is actor-name) {

Example owner-of coins is henry

owner-of cheese is nuked

Also see

Things to Remember:

• This only works on objects that have been "picked up" and are on the heap.

• Ownership may also be assigned to one of the following:
nobody:
It was picked up and is still on the heap but nobody currently has it in their inventory.

nuked:
Thrown off the heap.

in-the-room:
The object is in the room and not on the heap. The owner of the object must be in-the-room for pick-up-object to work.

• If you wish to take something out of the inventory, to be picked up again later, you first have to nuke it to remove the object code from the heap, and then set owner-of object to in-the-room. If the object is not nuked, it will still leave your inventory but when you pick it up again, you now have two, (and then three, then four...).

palette

This command is used to manipulate colors and palettes.

Syntax palette red green blue in-slot slot-name
palette foo in-slot bar
palette slot

Example palette 255 120 120 in-slot 30
palette da-slot in-slot crayon-color
palette room-alternate-palette

Also see palette transform, palette shadow, palette cycle speed, palette intensity

Things to Remember:

- Can be used to copy palette slots to others
- Can be used to select alternate palettes
- There is now a file called RGB.def that contains defines for many RGB colors.

palette cycle-speed

This command adjusts the color cycle speed of the specified cycle-slot. This is can also be used to turn color cycling off.

Syntax palette cycle-speed cycle-slot is speed

Example palette cycle-speed 2 is 0

Also see palette intensity, palette transform

Things to Remember:

- Look under Gradient in Dpaint to see the color cycle slots and speed settings.
- There are 16 color slots and the speed can go from 0 to 64.
- Because this hasn't been used or tested much, the speed and color slots may be off by one.
- This is usually used to stop color cycling by setting the speed to zero.

palette intensity

This command changes the intensity of a color or a group of colors in the palette. Can also be used to adjust the intensity level of the red, green and blue values independently.

Syntax palette intensity intensity-value in-slot slot to slot
palette intensity red green blue in-slot slot to slot

Example palette intensity 255 in-slot 17 to 120
palette intensity 255 100 120 in-slot 17 to 120

Also see palette, palette shadow, cycle speed, palette transform

Things to Remember:

- 255 is the normal level of intensity.
- Values greater than 255 increase the intensity. (lighten)
- Values less than 255 decrease the intensity. (darken)
- Each successive palette intensity command overrides the previous one.

palette transform

This command switches between alternate palettes.

Syntax palette transform palette-number \
[in-slot start-slot to end-slot] \
steps number-of-steps

palette palette-number

Example palette transform 1 in-slot 0 to 255 steps 20

palette 0

Also see

Things to Remember:

- Palette-number is the palette to transform to.
- Steps is the number of breaks that the transformation takes place over.
- If steps=1 then this is equivalent to :
  palette palette-number
- Alternate palettes are defined in Flem under Set Transparent Color.

password-string

System variable.

Syntax

Example

Also see

Things to Remember:

pause-key

System variable. Key used for game pause.

Syntax

Example

Also see

Things to Remember:

pick

This function picks a number from a list based on an index.

Syntax var = pick expression of [expression...]
var = pick expression of [expression... default
expression]

Example start-animation bob pick random 2 from [jump kick spin]

state-of big-door is pick door-index \
from [open wide-open default closed]

Also see

Things to Remember:

- The pick function can be used in any complex expression.

- The optional default causes pick to return the following number if the index is out of range. Beware that if there is no default statement, an out of range pick will result in a scumm error.

pick-up-object

This command picks up an object, puts it into the selected-actor's inventory (sets ownership), automatically sets the state of the object on the screen to GONE, and makes it UNTOUCHABLE.

Syntax pick-up-object object-name

Example pick-up-object phone

Also see

Things to Remember:

• If you wish to take something out of the inventory, to be picked up again later, you first have to nuke it to remove the object code from the heap, and then set owner-of object to in-the-room. If the object is not nuked, it will still leave your inventory but when you pick it up again, you now have two, (and then three, then four...).

• If you wish to pick up an object for an actor other than the selected-actor then use the following:

```
pick-up-object cheese
owner-of cheese is mouse
```

• Pick-up-object only works on objects who’s owner is set to in-the-room.

pixel

This function returns the value of the pixel at the given SCREEN coords

Syntax var= pixel x-screen-coord, y-screen-coord

Example foo = pixel 20, 20

Also see

Things to Remember:

print-cursor

This command prints text to the current cursor image

Syntax print-cursor [at x-coord, y-coord]
["string"]

Example print-cursor "skeletonized hand"

Also see

Things to Remember:

* You can also print a variable's contents.

print-debug

This command prints a string to the windex debugging monitor.

Syntax print-debug [at x-coord, y-coord]
["string"]

Example print-debug "driver paused"

Also see

Things to Remember:

* You can also print a variable's contents.

• Because this is on the windex screen -- not the VGA screen -- x-coord and y-coord refer to TEXT coordinates not pixels.

print-line

This command prints a string to the screen. Used for messages from non-actors (a PA system or when there are no actors present.).

Syntax print-line [left]
[overhead]
[color color-name]
[center]
[at x-coord, y-coord]
["string"[,]][+]][:]][-]]

Example
```
print-line  "Later That Day..."
print-line  left
print-line  "       Har Har Har"+
print-line  color light-blue "Har Har"
print-line  overhead
print-line  at 160,40
print-line  center
```

Also see say-line, wait-for-message, charset

Things to Remember:

- An actor's talk animation is not activated as it is with say-line.

- The left, color, overhead and center parts of the statement set the print-line default in preparation for a string to print to the screen. If no string is specified, the system will apply the default to all subsequent print-lines until the defaults are changed.

DON'T BELIEVE ANYTHING BELOW THIS POINT!!! print system being revamped!

- There are a few characters that can be embedded in messages. It the charset does not follow the standard SCUMM charset conventions these won't work. See the ASCII table in the Appendix for guideline on the SCUMM charset conventions:
  ^ will produce ... (three dots)
  @ is a non printing character
  ` will produce "

- Placing a "+" at the end of a print-line will keep the message on the screen indefinitely awaiting a further print-line, which is printed next to the first. beware of complication caused by using center in conjunction with this.

- Placing a "-" at the end of a print-line will add the next print-line directly after. This is used in conjunction with auto wrapping text. The next line must be a string.

- Placing a ":" at the end of a print-line will cause the lines to be printed after each other with a timed pause. This is short for wait-for-message. Only about three line can be put together in this way.

- Placing a "," at the end of the print-line will cause the 2nd line to be printed below the first like a carriage return.

- Print-line will erase any previous print-lines or say-lines but not print-texts. To erase the text, you need at least one space in the quotes. Also print-line " " will shut everybody's mouth. This should be done before special case animations, and after cut-scene overrides.

- If a message is pending on a "+", it takes one print-line " " to finish it and a second printline " " to get rid of it.

- It is possible to string print-lines together by using the "+" character. The following is taken from line 353 within Bardialog.scu in the game Monkey 1. The plus signs allow the 3 pirates to all laugh at the same time with the "Har Har's" all on one line.

Example:

```
  for j = 1 to 3 ++ {
      print-line left at 1,30 color light-blue "Har Har Har"+
      print-line left color yellow "       Har Har Har"+
      print-line left color light-purple "       Har Har Har"
      break-here 3
      print-line " "
      break-here
  }
```

• The color of the message can be optionally specified. The default color is white.

• print-line will wait for the camera to stop moving before printing.

* %o to imbed an object name
* %v to imbed a verb
* %s to imbed a string
* %n to imbed a scumm variable's contents
* %x in VERBS ONLY will perform a carriage return
* %x in print/say-line will have no affect, and it will remove all spaces immediately following. This is necessary for dialog interfaces that have more than 1 line.
* %f can be used to change charsets(fonts) DISABLED
* %c can be use to change color DISABLED

print-system

This command prints a string to the message box (ala pause).

Syntax print-system [color color-number]
["string"]

Example print-system color 154 "Do you want to quit? Y or N."

Also see

Things to Remember:

- It is possible for print-system to return a value. It returns the value of the key that was pressed to the system variable complex-temp.

* It can print the contents of a variable.

print-system-key-pressed

System variable set after calling 'print-system'

Syntax

Example

Also see

Things to Remember:

project.ifo

Contains a list of defines, object templates, room templates and the list of the rooms in the project.

Syntax

Example

Also see

Things to Remember:

proximity

This function returns the distance between two actors/objects or two points.

Syntax var=proximity actor-name actor-name
var=proximity actor-name to object-name
var=proximity x , y to x , y

Example break-until (proximity stan selected-actor < 100)

turn-threshhold = ((proximity act special-effect) / 2)

Also see

Things to Remember:

- Was using a cheap proximity routine, now returns true distance.
- If actor is scaled, proximity will be scaled as well.

put-actor

This command puts an actor at a specified location.

Syntax put-actor actor-name [at x-coord,y-coord]
[in-room room-name]
[at object-name]
[in-the-void]

Example
put-actor indy at 438,113
put-actor grail-knight at 192,110 in-room grail-chamber
put-actor radioman at radio
put-actor mouse in-the-void

Also see stop-actor, wait-for-actor, walk

Things to Remember:

- The put-actor at x-coord, y-coord command will put the actor at the coordinates given in whatever room they are in at that moment. Be careful, if the coordinates you select are not within a walk box, if the actor is following boxes, it will be placed at the edge of the nearest walkbox.

- Adding the parameter in-room will move the actor to the new room, and then place it at the indicated coordinates. However using in-room when the actor is already in the room will slow the system down.

- Putting an actor at an object places the actor at the "use-position" of the object. Objects are not valid unless they are in the current-room. Therefore the actor must already be in the room with the object.

- When you want to move an actor out of a current-room, but don't want to put it in any other room yet, you can put it into a room called in-the-void. If the actor is performing an animation when you used put-actor, it will put the actor at the new location and continue the animation, unless the actor was moved to a new room.

- When using the at-object parameter, the actor will automatically face in the direction of the object's use point.

- If the camera is already following an actor that you then move to a new room, the camera will pan to the actor's new x-coord. You must issue a new camera command to avoid this.

put-cursor

Places the cursor at the given screen coordinates

Syntax put-cursor x,y

Example put-cursor 320, 200

Also see cursor, cursor-room-x, cursor-screen-x

Things to Remember:

quit

This command exits the game.

Syntax quit

Example
```
if (copy-tries >= 3) {
  jump bail-out-of-the-game
}
bail-out-of-the-game:
  quit
```

Also see restart

Things to Remember:

- The quit statement is only used for copy protection and exiting a game.

random

This function returns a random number from 0 to the number indicated

Syntax var=random number

Example
```
timer = random 60
fly-x = (random 160 * 2) ; even number from 0 to 320
fly-y = (random 100 + 40); number from 40 to 100
```

Also see random-between

Things to Remember:

- The sequence of random numbers is seeded by the date/time that the game is started.

random-between

This function returns a random number between two numbers.
Random value is inclusive of the end values

Syntax var = random-between lower-number to upper-number

Example foo = random-between 10 100

Also see random

Things to Remember:

restart

This command restarts the game from the beginning. It clears all variables, terminates all running scripts, empties the heap, and starts the boot-script over again.

Syntax restart

Example restart

Also see quit

Things to Remember:

- This may be used after the end credits while in a loop waiting for a key press.
- WARNING: It does NOT put all actors in the void.

restart-key

System variable. Key used for game restart.

Syntax

Example

Also see

Things to Remember:

return

Allows a script to return a value.

Syntax return variable

Example return value
return 7
return object-count

Also see

Things to Remember:

- NO BREAKS OR SLEEPS ARE ALLOWED IN FUNCTION SCRIPTS!!!

right-button-state

System variable. Scripts can get button position. 1 is down and 0 s up.

Syntax

Example

Also see

Things to Remember:

.roo

Contain compiled flem and art information.

Syntax

Example

Also see

Things to Remember:

roofile

Created by MAKEMAKE, it is used to show dependencies used by MAKE to determine if a room must be recompiled.

Syntax

Example

Also see

Things to Remember:

room-height

System variable. Height for the current room.

Syntax

Example

Also see

Things to Remember:

room-width

System variable. Width of the currentroom (imagemin)

Syntax

Example

Also see

Things to Remember:

run-script

This is a macro that starts a script and then breaks until the script is not running.

Syntax run-script [bak][rec] script [^1 ^2]

Example run-script tell-joke

Also see

Things to Remember:

save-game

This command saves the state of the game to the heap after the next frame.

Syntax save-game

Example save-game

Also see save-load-key, load-game, saveload-enter-script, saveload-exit-script.

Things to Remember:

- Sounds are always saved, but you can control if sound is loaded during load-game.
- In the future, we may want to add the ability to select a slot to load into

save-load-colors

System variable. The colors for the control panel are in here.

Syntax

Example

Also see

Things to Remember:

save-load-enter-script

System variable. Script that gets run before each saveload.

Syntax

Example

Also see

Things to Remember:

save-load-exit-script

System variable. Script that gets run after each saveload.

Syntax

Example

Also see

Things to Remember:

save-load-key

System variable. Key used for savelaod screen. 0 for no savelaod.

Syntax

Example

Also see

Things to Remember:

saveload-enter-script

System variable. The script that gets run at start of saveload.

Syntax

Example

Also see

Things to Remember:

saveload-error

System variable. Saveload error flag for autosave and autoload.

Syntax

Example

Also see

Things to Remember:

say-line

This command prints a line to the screen and causes the speaker's mouth to animate.

Syntax say-line [actor-name] ["string"[[,]|[+]][:]
[center]
[at x-coord,y-coord]
[mumble]

Example say-line dr-fred "Leaping labrats!"
wait-for-message

say-line dr-fred mumble "sigh"

Also see print-line, wait-for-message, actor

Things to Remember:

• If an actor name is omitted, then the selected-actor speaks.

• Each actor can be assigned a text color using the actor command.

• Print-line " " will shut everybody's mouth. This should be done before special case animations, and after cut-scene overrides.

• Say-line will wait for the camera to stop moving before printing.

• Adding mumble to the say-line line causes the actor to speak with out calling the talk-animation.

* The contents of a variable can be printed.

* %f can be used to change charsets(fonts) DISABLED
* %c can be use to change color DISABLED

• Clipped is used to set the area of the screen that the say-line may print to by forcing it to wrap.

sayline-override-key

System variable. Key for overriding say-lines.

Syntax

Example

Also see

Things to Remember:

script-running

This function returns TRUE if the specified script is currently running.

Syntax var = script-running script-name

Example
```
if (not script-running practice-boxing) {
 start-script bak boxer-in-ring
}
if (script-running walk-dock-keeper) {
 load-lock-script meeting-of-the-masters
}
```

Also see start-script, stop-script, freeze-script, unfreeze-script

Things to Remember:

script-version-string

System variable. Version of game to be displayed when 'display-version-key' hit.

Syntax

Example

Also see

Things to Remember:

scripts.def

Every global script must be defined in this file. You can force the system to create this file by running MAKEDEFS.

Syntax

Example

Also see

Things to Remember:

.SCU

The SCUMM file. This file contains all of the code.

Syntax

Example

Also see

Things to Remember:

scumm.exe

The compiler for the SCUMM language.

Syntax

Example

Also see

Things to Remember:

scumm.ini

Sets the environment variables for the project. This makes it possible to have several project environments on one computer and simply switch between them. The environment variables set the paths for the project.

Syntax

Example

Also see

Things to Remember:

selected-actor

System variable used to hold who the current actor is.

Syntax

Example

Also see

Things to Remember:

selected-room

System variable which holds the current room number.

Syntax

Example

Also see

Things to Remember:

sentence-script

System variable. Script to run from do-sentence.

Syntax

Example

Also see

Things to Remember:

set-box

This command allows the programmer to change the status of one or more boxes.

Syntax set-box box-number [box-numer ...] to box-status

Example set-box 5 to box-normal
set-box 5 7 8 to box-invisible

Also see

Things to Remember:

There are a number of different types of box status. These are:

box-normal:
The normal state of a box. You would use this class to return a changed box to the norm.

box-flip-x:.
Normally when an actor is walking to the right it is facing right but if you say flip-walk-x the actor will actually walk to the left and do a little moon walk

box-flip-y:
Normally the camera looks down on a screen and the actor coming toward the screen is forward and going away from it is back. However in some instances, such as going behind hills, it is desirable to change this.

box-player-only [invisible] [locked]:
box-player-only is used as a prefix for the box classes box-player-only-invisible and box-player-only-locked. This is used in conjunction with the actor class player-only. (In Monkey, box-player-only was used in the store. Guybrush is set to the class "player-only" when he enters the store, and then the boxes behind the counter are set to box-player-only-invisible, which indicates that this box is invisible only to actors who are class "player-only.")

box-invisible:
Probably the most useful class, this pretends that a box doesn't exist. Setting a box invisible doesn't remove it from the connectivity matrix. It is possible to walk through an invisible box on your way to a visible box, but you cannot stop in the invisible box, nor get to the invisible box by clicking directly on it.

dirLB, dirFB, dirL, dirR, dirF, dirB, dirALL:
Sets the direction that the actor can walk in the box. “L” stands for left, “B” stands for back, etc. dirALL set the box back to the default setting allowing walking in all directions.

ADD ANGLE-OVERRIDE?

ONE-WAY
TWO-WAY?

set-box-path

This command rebuilds the box-connectivity table.

Syntax set-box-path

Example set-box-path

Also see

Things to Remember:

- Most commonly used after you've set a bridge box between two other boxes to box-invisible.
- This may cause a noticeable slow down in rooms with alot of boxes. Avoid using this statement when actors are moving.

set-box-set

THIS COMMAND NEEDS DOCUMENTATION.

Syntax set-box-set set

Example

Also see

Things to Remember:

sets

Groups of numbers that can be tested against. These can be represented as actors [guybrush,elaine,wally], objects [door, chair, window] or any series of numbers [1,3,5,7]

Syntax

Example

Also see if in, for

Things to Remember:

sleep-for

Causes a break measured in minutes, seconds or jiffies.

Syntax sleep-for number [jiffy|jiffies]
[seconds|second]
[minutes|minute]

Example sleep-for 3 seconds
sleep-for 30 jiffies
sleep-for (timer + 10) seconds
sleep-for random-between 2 to 4 seconds

Also see break-here, break-until

Things to Remember:

• Sleep-for is not sensitive to machine speed. Sleeping time is based on real clock time not frames rate.

• Sleep-for should not be used when an exact number of frames must pass -- such as during an animation sequence. Break-here should be used when timing animations.

• It is possible to say sleep-for random jiffies.

• Scumm internally converts all units to jiffies.

• 60 jiffies is equivalent to 1 second.

.sou

Sound and music effect files.

Syntax

Example

Also see

Things to Remember:

sound

A command hook for imuse commands.
Do not call directly. Use macros in imuse.def.

Syntax sound param [param...]

Example set-priority sound prio
set-vol sound vol
set-pan sound pan
fade-vol sound vol jiffies
fade-pan sound pan jiffies

Also see

Things to Remember:

sound-running

This function returns TRUE if the specified sound is currently running.

Syntax var = sound-running sound-name

Example
```
if (not sound-running machine-gun) {
   start-sound machine-gun
}
```

Also see

Things to Remember:

- This will give the status of either a sfx or of music.

special case animation

Animations, usually choreographed in CYST, that allow an actor to do something other than walk and talk. These animations are usually only use once in a game.

Syntax

Example

Also see

Things to Remember:

sputm-debug

System variable. SPUTMDEBUG environment variable allows ^g jumps.

Syntax

Example

Also see

Things to Remember:

sputm.exe

The interpreter for the SCUMM language.

Syntax

Example

Also see

Things to Remember:

stamp-actor

This command draws an actor into the room.

Syntax stamp-actor actor at x, y [scale scaleVal]

Example stamp-actor hoagie at 20, 20

stamp-actor bobbin at 50, 50 scale 100

Also see

Things to Remember:

- If an object changes states beneath this actor, or the room scrolls, actors drawn into the background will be erased.

stamp-object

THIS COMMAND NEEDS DOCUMENTATION.

Syntax

Example

Also see

Things to Remember:

start-music

This command begins playing a piece of music.

Syntax start-music music-name

Example start-music throttle-theme

Also see start-sfx

Things to Remember:

- Pieces started with this command are grouped as music by iMuse. This gives you separate volume control over voice, sfx, and music.

start-object

This command launches the code within an object/verb immediately, rather than having the actor walk over to the object first as part of a do-sentence execution. It can also be use as a function.

Syntax start-object [bak][rec] object-name verb verb-name
[expression [,] ...]

when used as a function:
var = start-object object-name verb verb-name
var = (start-object object verb verb [^1 ^2 ...)

Example start-object jungle-exit verb walk-to

when used as a function:
var = start-object current-noun1 verb get-icon

The verb would look like this:

```
verb get-icon {
   return bucket-of-icon
}
```

Also see

Things to Remember:

- This statement is particularly useful when there are no actors in a room or when you want to have an actor do the same thing from different places on the screen.
- To make typing code easier, @ can be used to replace start-script or start-object
- Be very careful when calling objects as functions. Function calls must match return statements exactly or major stack problems ensue and the game may crash.

start-script

This command tarts the execution of a new script, automatically loading it onto the heap if it's not already there. It can be run as a function.

Syntax [start-script] [bak][rec] script-name [(] [^1 ^2] [)]

foo = start-script script-name
foo = (start-script ^1 ^2)

Example start-script waking-up-fred decaf-coffee

foo = start-script find-dr-fred
foo = (start-script get-my-number indy)

Also see stop-script, chain-script, freeze-script, unfreeze-script, cut-scene, dialog, script

Things to Remember:

• Besides the normal way, a script can use two special commands:
BAK
This is a background script which is unaffected by the freezing of scripts. So these scripts will be running while cutscenes are occurring. Use for background animation (fire flickering) or special "watchdog" scripts, like waiting for an actor to arrive at a specific coordinate.
REC
This is a recursive script which can have multiple copies of itself running at the same time. If you have copies of a script running with the rec command, and then launch yet another copy but without rec, all copies will stop and just the one will run.

• Remember that the system can only have 20 scripts running at one time.

• The term start-script is optional. Start-script and Start-object can also be replaced by @.

• Start-script can accept parameters. In the example above, waking-up-fred is the script name and decaf-coffee is a variable being passed to the script. When the script was written it looked like this:

```
script waking-up-fred which-coffee {
        if (which-coffee is decaf) {
                < do a bunch of stuff >
        }
        if (which-coffee is caf-coffee {
                < do a bunch of different stuff >
        }
}
```

• When calling scripts as functions, the last statement of the script is a return statement which specifies a value to be placed in the variable.

• Function scripts must NEVER SLEEP or BREAK. And they must never run in the background or recursive. (As they never sleep or break there should never be a need to.)

* Function scripts must terminate or the machine will hang.

• When passing parameter to a function script, you must group the script call in parentheses because start-script is a complex expression.

start-sfx

This command begins playing a sound effect

Syntax start-sfx effect-name

Example start-sfx explosion

Also see start-music

Things to Remember:

- Pieces started with this command are grouped as sfx by iMuse. This gives you separate volume control over voice, sfx, and music.

start-variables

Prefaces a list of variables that are local to a room. This tells the system how to start numbering the variables that follow.

Syntax start-variables [expression] [room-locals]

Example start-variables room-locals

Also see

Things to Remember:

- This must be used if there are to be local variables in a room.

start-video

This command begins playing a video sequence

Syntax start-video "video name"

Example start-video "intro.san"

Also see

Things to Remember:

- Video files will be played from VIDPATH.

state-of

This command sets the state of an object. Can also be used as a function.

Syntax state-of object-name is state

```
var = state-of object-name

if (state-of object-name [is, is not] state-name) {
   [statements]
}
```

Example

```
state-of kitchen-entryway-door is CLOSED

foo = state-of kitchen-cabinet-door

if (state-of door is OPEN) {
   say-line "I should close this."
}
```

Also see

Things to Remember:

- There are up to 16 states. The background is 0.
- Only one state can be set within each state-of command. Use multiple commands to set states of additional states.
- State-of can set the state of an object regardless of the object's location. The object does not have to be in the current room.
- The following are the settings of state 0.
  HERE, CLOSED, R-OPEN (reverse-open which is same as CLOSED), OFF, R-GONE (reverse-gone which is same as HERE)
- The following are examples of setting the state to 1.
  OPEN, GONE, R-CLOSED (reverse-closed which is same as OPEN), ON, R-HERE (reverse-here which is same as GONE)
- State-of is very slow, because state-of deals with the proper rules of displaying objects on the screen. They may be completely hidden by another object or they could be overlapping another object. For example, you open a cabinet with a flashlight inside. Pick up the flashlight and when you close the cabinet the system will not try to draw the flashlight on top of the cabinet.
- Draw-object just blasts the image onto the screen background. It does not matter what was there before. Animation sequences that are constantly moving are ideal for draw object and not state-of. Use draw-object when you don't care about the rules and just want the image drawn as fast as possible.

stop-actor

Used to stop an actor from walking.
This is just a macro short-cut for 'actor <foo> stop' in scummmac.def

Syntax stop-actor actor-name

Example stop-actor indy

Also see actor

Things to Remember:

- defined in scummmac.def file.
- This automatically triggers do-animation stand and clears the walking flag.
- This is most useful when overriding a cut-scene where characters may be walking around. Usually put-actor is used to place the actor into place. However put-actor does not effect animation. If put-actor is used and the actor was walking when the override was hit, the actor would continue the walking animation in place. Stop-actor will make the actor stop walking.
- Stops actor at the next frame.

stop-object

This command stops the object's verb code.

Syntax stop-object object-name

Example stop-object closet-door

Also see start-object, stop-script

Things to Remember:

stop-script

This command stops the execution of a script.

Syntax stop-script [script-name]

Example stop-script tentacle-chases-kid

Also see start-script, freeze-script, unfreeze-script, cut-scene

Things to Remember:

- If the name of the script is omitted, then the current script is stopped.
- Scripts can be stopped at any point by this command. Therefore, it may be necessary to reset graphic and other states in conjunction with this.
- As all cutscenes must end through their end bracket, it is important to not use stop-scripts in the middle of cutscenes.

stop-sentence

This command terminates the sentence script and clears the do-sentence stack.

Syntax stop-sentence

Example
```
script move-and-get-clobbered {
   break-until (actor-moving selected-actor)
   stop-actor selected-actor
   stop-sentence
   hit-guybrush-again
}
```

Also see

Things to Remember:

stop-sound

This command stops a sound.

Syntax stop-sound sound-name

Example stop-sound kaboom

Also see

Things to Remember:

- Stops the sound immediately. It does not wait until the next frame.
- Use the iMuse version of stop-sound to have execution at the beginning of next frame.

string-width

This function returns the width in pixels of the specified string

Syntax var = string-width charset foo "string"

Example width = string-width charset fat-font "hello world"

Also see

Things to Remember:

system variable

Variables used by the system. They are declared and commented in SCUMMSYS.DEF

Syntax

Example

Also see

Things to Remember:

undim

This command deallocates the space on the heap that was reserved for an array.

Syntax undim variable

Example undim maze-map

Also see dim, arrays

Things to Remember:

- It is legal to use local variables in arrays. However, they must be UNDIMed before the script ends or the pointer will be lost and the space for the array on the heap can not be used again. You can now use 'dim array local' to have these arrays automatically deleted.

unfreeze-scripts

This command unfreezes all scripts. It is called by the cut-scene statement system level scripts, enter and exit scripts and dialogs.

Syntax unfreeze-scripts

Example unfreeze-scripts

Also see start-script, stop-script, cut-scene, dialog, enter code, exit code

Things to Remember:

- It is not usually necessary to use this command in everyday scripting.

unlock-

This command unlocks the item on the heap but doesn't remove it.

Syntax

```
unlock-costume    costume-name
unlock-room       room-name
unlock-script  global-script-name
unlock-sound   midi-file
```

Example

```
unlock-costume  nurse-edna
unlock-room     ednas-room
unlock-script  edna-chases-kid
unlock-sound  kid-screams
unlock-music  lechuck-theme
```

Also see

Things to Remember:

- The next time space is needed on the heap for something, the item might then be removed, as long as it's not being used in the current-room.

update-inven-script

System variable. Script to update inventory.

Syntax

Example

Also see

Things to Remember:

userput

This command activates and deactivates the keyboard and mouse.

Syntax userput on|off|soft-on|soft-off

Example userput on
userput off
userput soft-on
userput soft-off

Also see

Things to Remember:

• This command is used in the system level scripts, cutscenes,enter and exit scripts, dialogs and to set up unusual interfaces.

• When manipulating the userput it is almost always necessary to manipulate the cursor.

• on = set to 1, off = set to 0, soft-on = increments, soft-off = decrement

• Soft-on and soft-off can be the source of subtle bugs. Be very cautious when using this command. Always be sure that for every soft- off there is a matching soft-on and vise versa.

• build-sentence-script is ONLY run if userput > 0

userput-state

System variable.

Syntax

Example

Also see

Things to Remember:

valid-verb

This function returns TRUE if the verb listed is within the object's definition, otherwise it returns FALSE.

Syntax var=valid-verb object-name, verb-name

Example if (valid-verb current-noun1,give) {

Also see

Things to Remember:

variable

Declares a variable.

Syntax variable variable-name [number] [number] [] [...]

variable variable-name[array-size]

Example variable maid-x maid-y

variable actor-stuck[15]

Also see array, dim

Things to Remember:

- Compile time arrays are defined in the variable statement. The variable holds the pointer to the array on the heap.

variables.def

All global variables are declared here. All new variables must be added to the end of the appropriate list.

Syntax

Example

Also see

Things to Remember:

verb

The verb statement controls the look of the interface.

HUH?

Syntax

```
verb verb-name    [at x-coord,y-coord]
  [new]
  [color color-name]
  [dimcolor color-name]
  [hicolor color]
  [key key-name]
  [on]
  [off]
  [dim]
  [name "string"]
  [image object-number
  [bakcolor color]
  [center]
```

Example

```
verb open at 100,171    \
  new                \
  color green        \
  hicolor yellow     \
  dimcolor dark-grey  \
  bakcolor light-grey \
  key key-f          \
  on                \
  off               \
  dim               \
  name "turn on"     \
  image 125 \
  center
```

Also see

Things to Remember:

• The verbs are initialized in the main-scripts.scu and can be updated during game play.

• Here are the individual components of the verb statement:

at
Allows you to place a verb at an x and y-coord as the x and y position of the verb.

new
This needs to be used when a verb is first initialized.

color
This is the regular color of the verb. The default color is green.

dimcolor
Dimcolor is used to show a verb on the screen even though it is untouchable. The default dimcolor in is dark grey.

hicolor
Activated when the cursor moves over a "touchable" verb. The default hicolor is yellow. The verb automatically returns to its normal color when the cursor is off the verb.

bakcolor
Set the color that the verb is erased with.

key
Sets the keyboard key that is equivalent to a mouse click on the verb.

on/off/dim
These set the visual state of the verb. Off removes it from the screen, dim leaves it on the screen, in the dimcolor, but "untouchable", and On activates the verb.

name
This is the name the user will see on the screen when the verb is visible. It also allows you to change the verb's name.

center
Centers the text on the button.

image
Verbs can be either text or an image. If it is an image, this tells where to get the image. Object-number refers to the object number in flem.

• A maximum of 255 verbs can be associated with any given object. Thought this may change depending on the project.

• Multiple verbs can be used within a single verb definition.

verb-delete

This command kills the whole definition of the verb from the screen and the heap.

Syntax verb-delete verb-name

verbs-delete from-verb to to-verb set verb-set

Example verb-delete travel-1

Also see verbs-save verbs-restore

Things to Remember:

- verb-delete kills the whole definition of the verb. If it only needs to be of the screen for a little while, use verb off.

- These commands are so infrequently used, that we should consider removing them.

verb-x

This function returns the x position of the specified verb in SCREEN coords

Syntax var = verb-x verb-name

Example x-pos = verb-x open

Also see

Things to Remember:

verb-y

This function returns the y position of the specified verb in SCREEN coords

Syntax var = verb-y verb-name

Example y-pos = verb-y open

Also see

Things to Remember:

verbs-delete

THIS COMMAND NEEDS DOCUMENTATION.

Syntax verbs-delete from-verb to to-verb set verb-set

Example

Also see verbs-delete, verb-restore

Things to Remember:

verbs-restore

THIS COMMAND NEEDS DOCUMENTATION.

Syntax verbs-restore from-verb to to-verb set verb-set

Example

Also see save-verbs-save, verbs-delete

Things to Remember:

verbs-save

THIS COMMAND NEEDS DOCUMENTATION.

Syntax verbs-save from-verb to to-verb set verb-set

Example

Also see verbs-delete, verb-restore

Things to Remember:

video-speed

System variable. Speed of copies to video.

Syntax

Example

Also see

Things to Remember:

wait-for-actor

This command waits before running the next line of code until the actor gets to its destination.

Syntax wait-for-actor actor-name

Example walk indy to 100,120
wait-for-actor indy

Also see actor-moving, walk, do-animation

Things to Remember:

- This works with walking and turning only -- not for animations.

* All wait commands break the script they are running but they continue to run all other scripts.

wait-for-animation

This command waits before running the next line of code until the actor finishes an animation.

Syntax wait-for-animation actor-name

Example
```
do-animation indy whip-hook
wait-for-animation indy
```

Also see actor-moving, walk, do-animation

Things to Remember:

* All wait commands break the script they are running but they continue to run all other scripts.

* An animation is assumed to be completed when a cel doesn't change.

wait-for-camera

This command waits before processing the next line of code until the camera has stopped moving.

Syntax wait-for-camera

Example
```
camera-pan-to 320
wait-for-camera
```

Also see camera-follow, camera-pan-to, camera-at

Things to Remember:

- This statement is nearly always used with the camera-pan-to statement to make sure that the camera has reached its destination before the resumption of code processing.

* All wait commands break the script they are running but they continue to run all other scripts.

wait-for-message

This command causes a break-until the current message has finished being displayed.

Syntax wait-for-message

Example say-line sandy "My friends will save me!"
wait-for-message

Also see say-line, print-line, print-text, break-here, sleep-for, break-until

Things to Remember:

- Wait-for-message is equivalent to:
  break-until (message-going == false)
  (* message-going is a scumm system variable)

* All wait commands break the script they are running but they continue to run all other scripts.

wait-for-sentence

This command waits until all do-sentence's in the stack have been executed.

Syntax wait-for-sentence

Example do-sentence walk-to crossing-near-post
wait-for-sentence

Also see do-sentence

Things to Remember:

* All wait commands break the script they are running but they continue to run all other scripts.

wait-for-turn

This command waits before running the next line of code until the actor finishes turning.

Syntax wait-for-turn actor-name

Example
```
actor-foo face NORTH
wait-for-turn indy
```

Also see actor-moving, walk, do-animation

Things to Remember:

* All wait commands break the script they are running but they continue to run all other scripts.

walk

This command instructs an actor to walk to a specific location in the room. This will turn the actor to face the direction of the walk destination and then walk the actor to that location.

Syntax walk actor-name to [x-coord,y-coord]
[actor-name [within number]]
[object-name]

Example walk bernard to 473,107
walk indy to vogel within 24
walk indy to casket

Also see use-position, wait-for-actor,

Things to Remember:

- Adding the actor parameter walks an actor to the current x and y coord position of another actor. If used in conjunction with within, this command tells the actor to walk within a certain number of pixels of the target actor.

- Walking an actor to an object walks an actor to the use position of a touchable object, and faces him in the direction indicated by the use-position. The actor must be in the same room as the object.

- Walk to actor doesn't keep checking the current position of the target actor, it walks the actor to the target actor's coordinates as they were at the time the command was executed. So if a chase is required, this command will have to be placed inside a loop.

Example:

```
do {
    walk weird-ed to eds-temp within 2
    break-here
} until (proximity(weird-ed, eds-temp) <= 2)
```

walk talk animation

Animations, usually done in CYST, that allow an actor to walk and talk.

Syntax

Example

Also see

Things to Remember:

while

While is similar to until, but the condition test is performed BEFORE any statements are executed.

Syntax

```
while (condition) {
        [statements]
}
```

Example

```
while (foo > 7) {
      <do a bunch of stuff>
}
```

Also see until

Things to Remember:

* It the while statement is NEVER true, the statements contained within it will never be executed.

.zb?

Zplane clipping files.

Syntax

Example

Also see

Things to Remember: