================================================================================

GENERAL DEFINITIONS

================================================================================

For Usecode C purposes, the avatar is an NPC just like any other. Constants

referred to below (usually in ALL CAPS) are defined in

'src/headers/constants.uc' unless otherwise stated.

Unlike Usecode C-defined functions, intrinsics do not have a 'prototype'; you

can pass as many (or as few) parameters to an intrinsic as you like. The only

'hard' limits are that Exult will not like if you (1) call an intrinsic with

more than 12 parameters or (2) the intrinsic expects parameters which you

haven't supplied. The parameter lists in the list below is the minimum safe

number of parameters.

To the right of many intrinsic 'prototypes', you can see an indicator between

square brackets. This indicator is a quick way to see if that intrinsic is

available for the specific game you are interested in. The possible values of

these indicators are:

Indicator Meaning

[BG] BG-specific intrinsic

[SI] SI-specific intrinsic

[Exult] Exult-specific intrinsic, available to both BG and SI

[BG, Exult] Originally BG-only, Exult allows it in SI too

[SI, Exult] Originally SI-only, Exult allows it in BG too

[Exult: BG] Exult-specific intrinsic, available only to BG

[Exult: SI] Exult-specific intrinsic, available only to SI

(not present) Intrinsic available for BG and SI

The '[Exult]', '[BG, Exult]' and '[SI, Exult]' indicators are essentially

equivalent to no indicator at all; they are used in this document only for

completeness.

The intrinsic prototypes below are in Usecode C-esque form. It is actually more

like C than Usecode C. Specifically, all parameters have a type instead of the

Usecode C-'var'. The parameter type indicates what Exult will be expecting; the

Usecode C compiler won't care in the slightest what you pass as a parameter. The

parameter types I use are as follows:

Keyword Description

int This means any integer, positive or negative.

object Any object or NPC. It can be an object reference, 'item' or a

number (that \*must\* be negative, which Exult will convert to an

NPC if it is in the correct range).

actor Any NPC. It is evaluated the same way as 'object' is, but Exult

will reject anything that can't be converted to an NPC. Monsters

are also accepted.

bool A 'true'/'false' parameter. Can be a number (nonzero/zero), an

array (non-empty/empty) or a comparison expression (e.g., 'a ==

b'.

string A string (text). Strings in usecode are any set of characters

surrounded by double-quotes; for example, "This is a string.".

If you need a double-quote inside a string you can 'escape' it

by prefixing them with a backslash: "\""; alternatively, Exult

converts '@' into a double-quote.

function A usecode function, which \*must\* be declared with either, the

shape# /or/ the object# specifiers. In Usecode C, this can be

the function name or its usecode number. For example: 'extern

Bolt shape#(0x2D3) ();' can be passed as 'Bolt' (the function

name), '0x2D3' (its function number since 'shape#(shnum) ==

shnum' if shnum < 0x400) or as '&Bolt' (using the 'address of'

operator '&' to get the function number).

If the parameter name is followed by a '[]', it indicates an array with elements

of the specified type. Example: 'actor npc[]' means an array of NPCs. If there

is a number between the brackets, it means an array of that length; for example,

'int num[3]' means an array with 3 'int' elements. In some cases, an array can

be of type 'mixed' to indicate that not all of the array's elements are of the

same type.

Some intrinsics are 'overloaded'; i.e., they accept more than one parameter

type. In those cases, all of the possibilities are listed as prototypes.

If the parameter list is simply '[special]', see the description for details.

If the intrinsic is preceded by a parameter type, it means that the intrinsic

returns something of that type; the description explains the return value. In

some cases, the return value can be 'mixed[]' to indicate that the intrinsic

returns an array, not all of which are of the same type.

All intrinsics can be called as presented in the list. If an intrinsic has at

least one parameter, it can be called in 'calle' form, with the first parameter

as its 'object'. Example:

int UI\_execute\_usecode\_array(object obj, int script[]);

int obj->execute\_usecode\_array(int script[]);

The first parameter and the '->' can both be omitted if 'obj' is 'item', and if

you omit either of them you \*must\* omit both or the Usecode will fail to

compile. For stylistic purposes, the 'calle' form should be used only when the

first parameter is an object or NPC (or 'item'). For all intrinsics with

'appropriate' parameter types, I have listed both forms in the list below.

Also, you may have noticed that I did not include the parameter type of 'obj'

when I gave the 'calle' form of execute\_usecode\_array; since I will only give

the 'calle' form when the parameter type is 'object' or 'actor', I feel that it

is unneeded.

================================================================================

RANDOM NUMBER INTRINSICS

================================================================================

These intrinsics are all related to random numbers and their generation.

int UI\_get\_random(int max)

Generates a random number, similar to rolling a dice of the given side.

Parameters:

max The maximum value of the random number. Should be positive.

Return:

Zero if max is zero; otherwise, a random number ranging from 1 to max.

int UI\_die\_roll(int min, int max)

Generates a random number in the specified range.

Parameters:

min The minimum value of the random number. Should be positive.

max The maximum value of the random number. Should be positive.

Return:

A random number ranging from min to max. If min > max, the two parameters

are reversed by Exult.

bool UI\_roll\_to\_win(int att, int def)

Performs a contest between two supplied values.

Parameters:

att What should be added to the die roll, improving the chances of

success.

def What should be subtracted from the die roll, reducing the chances of

success.

Return:

The contest is a roll of a 30-sided die: if the die comes up 30, the return

is true; if it comes up 1, the return is false. For all other values, the

attack value is added to the die roll and the defense value is subtracted

from the same roll. If the result is 15 or more, the attacker wins and the

return is true; otherwise, the attacker loses and the return is false.

================================================================================

SCRIPTING INTRINSICS

================================================================================

These intrinsics are used to animate characters from usecode. Some of these

are deprecated by script blocks, but all others are still useful at times.

int UI\_execute\_usecode\_array(object obj, int script[])

int obj->execute\_usecode\_array(int script[])

Deprecated; use Usecode C script blocks instead. Animates an object

according to the supplied script.

Parameters:

obj The object to be animated.

script The sequence of actions to be executed.

Return:

Always returns '1'.

int UI\_delayed\_execute\_usecode\_array(object obj, int script[], int delay)

int obj->delayed\_execute\_usecode\_array(int script[], int delay)

Deprecated; use Usecode C script blocks instead. Animates an object

according to the supplied script after a delay. Works exactly like

execute\_usecode\_array, but starts only after the specified delay.

Parameters:

obj The object to be animated.

script The sequence of actions to be executed.

delay The number of ticks that must elapse before the animation starts.

Return:

Always returns '1'.

bool UI\_in\_usecode(object obj)

bool obj->in\_usecode()

Checks if an object is currently executing any usecode scripts.

Parameters:

obj The object to be inspected.

Return:

true if there are any scripts for the specified object, false otherwise.

UI\_halt\_scheduled(object obj)

obj->halt\_scheduled()

Terminates usecode scripts that animate the supplied object. Scripts that

include the 'nohalt' command are not affected by this intrinsic.

Parameters:

obj The object whose scripts should be terminated.

UI\_begin\_casting\_mode(actor npc, int val) [Exult]

npc->begin\_casting\_mode(int val)

Makes an NPC display casting frames until the end of its next script.

Parameters:

npc The NPC that should display casting frames.

val The shape in 'SHAPES.VGA' to use as casting frames. If omitted, this

is assumed to be shape 859.

int UI\_set\_to\_attack(actor attacker, object target, int weapon)

int attacker->set\_to\_attack(object target, int weapon)

int UI\_set\_to\_attack(actor attacker, mixed[4] target, int weapon)

int attacker->set\_to\_attack(mixed[4] target, int weapon)

Schedules a NPC to attack the object with a given weapon. This intrinsic

does not animate the NPC in any way; the scheduled attack only emulates the

actual hit with the selected weapon. The actual attack is triggered by a

script; in an Usecode C script, use the 'attack' command to actually perform

the attack.

Parameters:

attacker Who is attacking.

target What is being attacked. Can be an object, NPC, or the return of

a click\_on\_item intrinsic call, even if a tile has been clicked.

weapon What shape in 'SHAPES.VGA' to use as the source of the weapon

information in the attack. It \*must\* specify a shape that has a

'weapon' tab in Exult Studio, or nothing will happen.

If the NPC is targeting a tile, the attack will only be meaningful if the

weapon causes an explosion.

Return:

true if attacking a valid target with a valid weapon, false otherwise.

bool UI\_path\_run\_usecode(int pos[], function fun, object obj, int event, int

simode)

Causes the avatar to walk to a location and execute the specified usecode

function when he/she arrives at the destination. If a path cannot be found

for the destination, no usecode will be executed unless a call to

set\_path\_failure intrinsic is made before returning from usecode.

The avatar can be prevented from reaching his/her destination simply by

moving away from the path; this will cancel the effects of the intrinsic,

and will cause the usecode function set by set\_path\_failure to be called. If

you want the path to be uninterruptible, the avatar's DONT\_MOVE flag must be

set (this is flag 22 in BG and 16 in SI).

The other party members will follow the avatar towards the destination.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In SI, path\_run\_usecode finds a free spot within 3 tiles of the position

specified by 'pos'; it also allows a rise/drop of 3 in the z direction.

\*\*\*\* END SI SPECIFIC \*\*\*\*

Parameters:

pos The location that the avatar should walk to. Can be a (x, y)

position or a (x, y, z) position. If the z coordinate is missing,

or if it is negative, it is assumed to be zero.

fun The usecode function to be called if the destination is reached.

obj The object that should be used as item if the usecode function is

called. If this parameter is zero, usecode will \*not be called\* at

all.

event The event that should be used if the usecode function is called.

simode This optional parameter can be used to control whether or not to

use the SI-specific behavior described above. It defaults to true

in SI and to false in BG.

Return:

If no path can be found to the destination, returns false; otherwise,

returns true.

UI\_si\_path\_run\_usecode(actor npc, int pos[], int event, [SI, Exult]

object obj, function fun, bool flag)

npc->si\_path\_run\_usecode(int pos[], int event, object obj, function fun, bool

flag)

Causes an NPC to walk to a location and execute the specified usecode

function when he/she arrives at the destination. If a path cannot be found

for the destination, no usecode will be executed unless a call to

set\_path\_failure intrinsic is made before returning from usecode.

The avatar can be prevented from reaching his/her destination simply by

moving away from the path; similarly, a scheduled event or scripted event

may prevent an NPC from reaching his/her destination. In these cases, the

effects of the intrinsic will be cancelled, and the usecode function set by

set\_path\_failure will be called. If you want the path to be uninterruptible,

the NPC's DONT\_MOVE flag must be set (this is flag 22 in BG and 16 in SI).

Parameters:

npc The NPC that will be affected by this intrinsic. If this isn't a

valid NPC, nothing will happen.

pos The location that the NPC should walk to. Can be a (x, y) position

or a (x, y, z) position. If the z coordinate is missing, or if it

is negative, it is assumed to be zero.

event The event that should be used if the usecode function is called.

obj The object that should be used as item if the usecode function is

called. If this parameter is zero, usecode will \*not be called\* at

all.

fun The usecode function to be called if the destination is reached.

flag Flag specifying whether or not the usecode function should always

be called. If false, it will be called only if the NPC can reach

its intended destination; if true, the specified usecode function

will be called even if the destination cannot be reached. The

set\_path\_failure intrinsic will override this flag in either case.

This failure usecode will use event 'SI\_PATH\_FAILURE'.

UI\_set\_path\_failure(function fun, object obj, int event)

Sets usecode to be called if an NPC fails to reach its destination. This

intrinsic must be used \*after\* a call to either, path\_run\_usecode or

si\_path\_run\_usecode intrinsics, and will affect only the NPC affected by

that call. If, for any reason, the NPC in question cannot reach the

destination, the usecode function assigned by this intrinsic will be run.

Parameters:

fun The usecode function to be called if the path fails.

obj The object that should be used as item if the usecode function is

called. If this parameter is zero, usecode will \*not be called\* at

all.

event The event that should be used if the usecode function is called.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In the original SI, this could never affect the failure event for

si\_path\_run\_usecode, which would always be 14. Exult does not suffer from

this limitation.

\*\*\*\* END SI SPECIFIC \*\*\*\*

bool UI\_in\_usecode\_path(actor npc)

bool npc->in\_usecode\_path()

Checks whether or not the actor is traversing a path due to path\_run\_usecode

or si\_path\_run\_usecode intrinsics.

Parameters:

npc The NPC to be checked.

Return:

true if the NPC is still walking a usecode path, false otherwise.

UI\_telekenesis(function fun)

Deprecated; stores a usecode function for special script behavior. The next

\*script\* call to the stored function (e.g., 'script item call fun;') will

have an event ID of 'DOUBLECLICK' instead of the standard 'SCRIPTED' event

ID used for scripts. Since Usecode C scripts allow calling a function with

an event ID of your choice (e.g., 'script item call fun, DOUBLECLICK;') it

is best to simply ignore this intrinsic.

Parameters:

fun The function to be stored.

================================================================================

CONVERSATION INTRINSICS

================================================================================

These intrinsics are used to display faces for conversations, storing and

recalling answers, etc.

UI\_show\_npc\_face(int face, int frnum)

UI\_show\_npc\_face(actor face, int frnum)

face->show\_npc\_face(int frnum)

Displays a face onscreen for conversations. The face is a shape from

'FACES.VGA'. This intrinsic finds a free slot to display the face in or uses

the second slot if there are no free slots.

Parameters:

face The face to be displayed. May be an integer or an NPC.

frnum The frame number of the face to be displayed.

If 'face' is an integer, the shape to be displayed is the absolute value of

this number. If this value is 0 or 356, the avatar's face will be displayed,

using the correct face for the avatar's skin and PETRA status flag.

If 'face' is an NPC, the Exult Studio-set face for that NPC is displayed;

the "default" face is equal to the NPC's number.

For the cases when 'frnum is zero, you should use Usecode C's 'npc.say()' to

display the NPC's face and display any text within the parenthesis (or set

with 'message') too.

UI\_show\_npc\_face0(int face, int frnum) [SI, Exult]

UI\_show\_npc\_face0(actor face, int frnum)

face->show\_npc\_face0(int frnum)

Displays a face onscreen for conversations. The face is a shape from

'FACES.VGA'. This intrinsic always uses slot 0 for the new face.

Parameters:

face The face to be displayed. May be an integer or an NPC.

frnum The frame number of the face to be displayed.

If 'face' is an integer, the shape to be displayed is the absolute value of

this number. If this value is 0 or 356, the avatar's face will be displayed,

using the correct face for the avatar's skin and PETRA status flag.

If 'face' is an NPC, the Exult Studio-set face for that NPC is displayed;

the "default" face is equal to the NPC's number.

For the cases when 'frnum is zero, you should use Usecode C's 'npc.say()' to

display the NPC's face and display any text within the parenthesis (or set

with 'message') too.

UI\_show\_npc\_face1(int face, int face) [SI, Exult]

UI\_show\_npc\_face1(actor face, int face)

npc->show\_npc\_face1(int face)

Displays a face onscreen for conversations. The face is a shape from

'FACES.VGA'. This intrinsic always uses slot 1 for the new face.

Parameters:

face The face to be displayed. May be an integer or an NPC.

frnum The frame number of the face to be displayed.

If 'face' is an integer, the shape to be displayed is the absolute value of

this number. If this value is 0 or 356, the avatar's face will be displayed,

using the correct face for the avatar's skin and PETRA status flag.

If 'face' is an NPC, the Exult Studio-set face for that NPC is displayed;

the "default" face is equal to the NPC's number.

For the cases when 'frnum is zero, you should use Usecode C's 'npc.say()' to

display the NPC's face and display any text within the parenthesis (or set

with 'message') too.

UI\_remove\_npc\_face(int face)

UI\_remove\_npc\_face(actor face)

face->remove\_npc\_face()

Deprecated; use Usecode C's 'npc.hide()' statement instead. Hides the

specified face if it is being shown.

Parameters:

face The face to be hidden. May be an integer or an NPC.

If 'face' is an integer, the shape to be hidden is the absolute value of

this number. If this value is 0 or 356, the avatar's face will be hidden.

If 'face' is an NPC, the Exult Studio-set face for that NPC is hidden; the

"default" face is equal to the NPC's number.

UI\_remove\_npc\_face0() [SI, Exult]

Removes whatever face is being shown in slot 0.

UI\_remove\_npc\_face1() [SI, Exult]

Removes whatever face is being shown in slot 1.

UI\_init\_reset\_conv\_face()

Changes the active conversation slot to the top slot and sets its current

frame to 0.

UI\_set\_conversation\_slot(int slot) [SI, Exult]

npc->set\_conversation\_slot()

Changes the active conversation slot.

Parameters:

slot The new active slot. Must be zero or one.

UI\_change\_npc\_face0(int frame) [SI, Exult]

Changes the frame of the face being displayed in slot 0.

Parameters:

frame The new frame to be used.

UI\_change\_npc\_face1(int frame) [SI, Exult]

Changes the frame of the face being displayed in slot 1.

Parameters:

frame The new frame to be used.

UI\_init\_conversation() [SI]

Initializes the face information for a conversation. In Exult, it is usually

safe to skip it.

UI\_end\_conversation() [SI]

Shows any pending text and removes all faces from the screen. Usually safe

to skip this in Exult.

UI\_add\_answer(string answer)

UI\_add\_answer(string answer[])

Deprecated; use Usecode C's 'add(string answer)' or 'add(string answer[])'

statements instead. Adds the specified strings as dialog options that the

player can select.

Parameters:

answer A string or array of strings containing the dialog options to be

added.

UI\_remove\_answer(string answer)

UI\_remove\_answer(string answer[])

Deprecated; use Usecode C's '(remove)' construct instead. Removes the

supplied strings from the dialog options available to the player.

Parameters:

answer A string or array of strings containing the dialog options to be

removed.

UI\_push\_answers()

Stores the current conversation options and clears all choices. The dialog

options are stored in a LIFO stack that persists until the dialog ends;

before that point, the stored options can be retrieved with a call to

UI\_pop\_answers intrinsic.

UI\_pop\_answers()

Clears all current conversation options and restores the stored conversation

options, if any. The restored dialog options are those stored by the /last/

call to UI\_push\_answers intrinsic.

UI\_clear\_answers()

Clears all current dialog options.

string UI\_select\_from\_menu()

Makes the player choose one answer from all dialog options currently being

displayed.

Return:

The chosen dialog option.

int UI\_select\_from\_menu2()

Makes the player choose one answer from all dialog options currently being

displayed.

Return:

The index (ranging from 1 to n) of the chosen dialog option.

================================================================================

SOUND INTRINSICS

================================================================================

All intrinsics in this section relate to playing sound effects, music or

speech.

UI\_play\_sound\_effect(int sfx)

Plays a sound effect.

Parameters:

sfx A numerical value indicating the sound effect that should be played.

UI\_play\_sound\_effect2(int sfx, object obj)

Causes an object to generate a sound effect.

Parameters:

sfx A numerical value indicating the sound effect that should be played.

obj The object generating the sound effect. The sound intensity decrease

as the avatar gets farther away from this object.

UI\_play\_music(int track, object obj)

Plays the specified song.

Parameters:

track The track number to be played or 255 (0xFF) to \*stop\* any music

that is playing.

obj The source of the music, or zero for 'unattached' music. Positive

numbers are \*not\* interpreted as NPCs!

If supplied with a valid object or NPC that is on the game world, this

intrinsic will also display the sprite effect # 24 (musical notes).

int UI\_get\_music\_track()

Gets the music being played.

Return:

If music is enabled and a song is playing, returns the track number of the

current song. Otherwise, return is -1.

\*\*\*\* NOTES ABOUT ORIGINAL BG \*\*\*\*

The return value in the original Black Gate was more complicated; the

current song was the return value divided by 256, while the next song (the

one that would be resumed after the current one ends) was the remainder of

that same division. None of this behavior is true for Exult, which works as

described above.

\*\* END NOTES ABOUT ORIGINAL BG \*\*

bool UI\_start\_speech(int track)

Plays the specified speech track.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In SI, speech also displays a full-screen face from 'FACES.VGA'.

\*\*\*\* END SI SPECIFIC \*\*\*\*

Parameters:

track Number indicating what speech track is to be played.

Return:

Returns false if failed to play the requested speech track (for example, if

speech is disabled), true otherwise.

bool UI\_start\_blocking\_speech(int track) [BG, Exult]

Plays the specified speech track. Blocks until the speech finishes.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In SI, speech also displays a full-screen face from 'FACES.VGA'.

\*\*\*\* END SI SPECIFIC \*\*\*\*

Parameters:

track Number indicating what speech track is to be played.

Return:

Returns false if failed to play the requested speech track (for example, if

speech is disabled), true otherwise.

int UI\_get\_speech\_track()

Determines the currently playing speech track.

Return:

A number indicating what voice track is being played.

================================================================================

SEARCHING AND COUNTING INTRINSICS

================================================================================

These intrinsics allow you to find and count objects, whether they are in a

container or in the game world.

object UI\_find\_nearest(object obj, int shape, int dist)

object obj->find\_nearest(int shape, int dist)

Searches the area around a given object looking for other objects of a

specified type.

Parameters:

obj The object around which the search will happen.

shape The shape we are looking for. Can be SHAPE\_ANY = -359 to match any

shape.

dist The maximum distance from 'obj' that will be considered in the

search.

Return:

An object reference to the nearest object found given the search criteria.

You cannot find eggs, barges, invisible or transparent objects with this

intrinsic.

object UI\_find\_object(object cont, int shape, int qual, int frame)

object cont->find\_object(int shape, int qual, int frame)

object UI\_find\_object(int cont, int shape, int qual, int frame)

object UI\_find\_object(int cont[3], int shape, int qual, int frame)

Finds the first object matching the search criteria.

Parameters:

cont The object or location to be searched. Can be a container or NPC, a

(x, y, z) location, PARTY = -357 to search the whole party or

FIND\_ON\_SCREEN = -359 to search a square region around the screen

location with size given by the screen's height.

shape The shape we are looking for.

qual The quality that is being sought. Can be QUALITY\_ANY = -359 to

match any quality.

frame What frame is desired. Can be FRAME\_ANY = -359 to match any frame.

\*\*\*\* IMPORTANT NOTES \*\*\*\*

When 'cont' is a position, the search is made in a 1-tile-wide region

centered on the given position.

Also, if the search is \*not\* being done in a container or NPC, 'shape' can

be SHAPE\_ANY = -359 to match any shape.

\*\* END IMPORTANT NOTES \*\*

Return:

An object reference to the first found object, or zero if no objects match

the given criteria.

object[] UI\_find\_nearby(object loc, int shape, int dist, int mask)

object[] loc->find\_nearby(int shape, int dist, int mask)

object[] UI\_find\_nearby(int loc[3], int shape, int dist, int mask)

object[] UI\_find\_nearby(int loc[5], int shape, int dist, int mask)

object[] UI\_find\_nearby(mixed loc[4], int shape, int dist, int mask)

Searches a specified area looking for objects matching a given criteria.

Parameters:

loc The center of the area to be searched. Can be the return of a

'UI\_click\_on\_item' intrinsic call, an object reference, a (x, y, z)

position vector or a (x, y, z, quality, frame) vector. For the last

form, quality can be QUALITY\_ANY = -359 and frame can be FRAME\_ANY

= -359 (but there is little point in both being wildcards, as this

is the default behavior).

shape The shape we are looking for. Can be SHAPE\_ANY = -359 to match any

shape.

dist Radius (in tiles) of the area to be searched.

mask Parameter specifying the classes of objects that can be found.

The mask value controls what can be found; using the proper mask, you can

find eggs, barges, invisible or transparent objects with this intrinsic. The

mask can be the combination (sum) of one or more of the following values:

MASK\_NONE = 0x00 Cannot find eggs, barges, invisible or

transparent objects.

MASK\_NPC = 0x04 Restricts the search to NPCs, alive or

dead. Ignored if a non-wildcard shape is

given.

MASK\_NPC2 = 0x08 Restricts the search to living NPCs.

MASK\_EGG = 0x10 Allows eggs and barges to be found.

MASK\_INVISIBLE = 0x20 Allows invisible objects that are not in

the avatar's party to be found.

MASK\_PARTY\_INVISIBLE = 0x40 Allows invisible party members to be

found

MASK\_TRANSLUCENT = 0x80 Allows translucent objects to be found

Return:

Array containing all objects of matching the given criteria that are located

in the specified area. This array is sorted from right to left, closest

first.

object[] UI\_find\_nearby\_avatar(int shape)

Searches the area around the avatar looking for other objects of the

specified shape.

Parameters:

shape The shape we are looking for. Can be SHAPE\_ANY = -359 to match any

shape.

Return:

Array containing all objects of the given shape that are located within 192

tiles of the avatar. This array is sorted from right to left, closest first.

You cannot find eggs, barges, invisible or transparent objects with this

intrinsic.

bool UI\_npc\_nearby(object obj)

bool obj->npc\_nearby()

Finds out if the desired object is on screen and can act. An NPC is

considered able to act if it is not asleep, paralyzed or dead; other objects

are always considered able to act. Note that, despite the name, this

intrinsic can also be used for objects.

Parameters:

obj The object or NPC to be checked.

Return:

true if the object is in the screen and can act, false otherwise or if the

supplied parameter is not an object or NPC.

bool UI\_npc\_nearby2(object obj) [SI]

bool obj->npc\_nearby2()

Finds out if the desired object is nearby. In this context, 'nearby' means

within 40 tiles of the avatar. Note that, despite the name, this intrinsic

is valid for all objects, not just NPCs.

Parameters:

obj The object to be checked.

Return:

true if the object is within 40 tiles of the avatar, false otherwise.

int UI\_count\_objects(object obj, int shape, int qual, int frame)

int obj->count\_objects(int shape, int qual, int frame)

Counts all objects matching a given search criteria that are contained in

the supplied object.

Parameters:

obj The object whose contents are to be searched. Can be a container,

an NPC or PARTY = -357 to search the whole party.

shape The shape we are looking for. Can be SHAPE\_ANY = -359 to match any

shape.

qual The quality that is being sought. Can be QUALITY\_ANY = -359 to

match any quality.

frame What frame is desired. Can be FRAME\_ANY = -359 to match any frame.

Return:

The number of objects that match the given criteria.

object[] UI\_get\_cont\_items(object obj, int shape, int qual, int frame)

object[] obj->get\_cont\_items(int shape, int qual, int frame)

Looks for all objects matching a given search criteria that are contained in

the supplied object.

Parameters:

obj The object whose contents are to be searched. Can be a container or

an NPC.

shape The shape we are looking for. Can be SHAPE\_ANY = -359 to match any

shape.

qual The quality that is being sought. Can be QUALITY\_ANY = -359 to

match any quality.

frame What frame is desired. Can be FRAME\_ANY = -359 to match any frame.

Return:

Array containing object references to all objects that match the given

criteria.

================================================================================

PLACEMENT INTRINSICS

================================================================================

With these intrinsics, you can create, move or delete objects or NPCs,

whether they are in the game world or in a container.

bool UI\_is\_not\_blocked(int pos[3], int shape, int frame)

Checks if there is enough room for a given shape in the specified location.

Parameters:

pos The location to be checked.

shape The shape in 'SHAPES.VGA' that is to be checked.

frame The frame of the given shape that should be checked.

Return:

If there is enough room in the given location for an object of the given

shape and frame, returns true; otherwise, returns false.

actor[] UI\_add\_party\_items(int count, int shape, int qual, int frame, bool temp)

Creates a specified amount of objects matching the given parameters and

gives them all to the party. Not all objects requested may be created; in

particular, this intrinsic is guaranteed to create only as many of the

requested objects as the party can hold.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

If the party cannot hold all requested objects, the remaining objects are

created and then placed in the ground nearby.

\*\*\*\* END SI SPECIFIC \*\*\*\*

Parameters:

count The amount of objects to be created.

shape The shape in 'SHAPES.VGA' that should be used for the created

objects.

qual The quality of the objects being created. Can be QUALITY\_ANY = -359

to create items of any quality.

frame What frame will be used for the created objects. Can be FRAME\_ANY =

-359 to create items of any frame.

temp Flag indicating whether the created items are temporary or not.

Return:

Returns an array of object references for all party members that received

any of the created objects.

\*\*\*\*\*\* BG SPECIFIC \*\*\*\*\*\*

The 'temp' flag applies to all objects created by this function.

\*\*\*\* END BG SPECIFIC \*\*\*\*

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

If any objects were added to the ground, the number of objects on the ground

is appended to the end of the return array. The 'temp' flag applies only to

those objects that were added to the floor.

\*\*\*\* END SI SPECIFIC \*\*\*\*

bool UI\_remove\_party\_items(int count, int shape, int qual, int frame)

bool UI\_remove\_party\_items(int count, int shape, int qual, int frame, bool unk)

Looks through the party's possessions and deletes a specified amount of

objects matching a given search criteria. This function is an all-or-nothing

function: it will delete all requested objects or none at all if it found

less objects than the amount that was requested for deletion.

Parameters:

count The amount of objects to be deleted. Can be QUANTITY\_ANY = -359 to

delete all objects that match the criteria.

shape The shape we are looking for.

qual The quality that is being sought. Can be QUALITY\_ANY = -359 to

match any quality.

frame What frame is desired. Can be FRAME\_ANY = -359 to match any frame.

unk Flag of unknown meaning. This parameter is ignored by Exult, and

can be omitted.

Return:

true if 'count' objects were found and deleted, false if no objects were

deleted.

object UI\_summon(int shape)

object UI\_summon(int shape, bool unk)

Creates a monster with shape 'shape'. The created monster has equipment as

defined in its equipment record, and has a 'GOOD' alignment. If the creation

is successful, the return value is a reference to the monster; otherwise,

the return is '0'. In the originals, there was an additional 'bool'

parameter, but it is unimplemented in Exult.

Creates a monster of the supplied shape. This monster has equipment as

defined in its equipment record, starts with an IN\_COMBAT schedule, and will

have the same alignment as item; for the cases in which item is not an NPC,

the summoned monster's alignment will be GOOD instead. The monster will be

placed within five tiles of the avatar.

Parameters:

shape The shape in 'SHAPES.VGA' to be used for the summoned monster. It

\*must\* have a 'monster' tab in Exult Studio.

unk A flag of unknown meaning. Exult ignores this flag, and it is safe

to omit it.

Return:

Zero if there is no free tile near the avatar or the specified shape is not

a valid monster shape; an object reference to the summoned monster

otherwise.

actor UI\_clone(actor npc)

actor npc->clone()

Creates a duplicate of the supplied NPC. This duplicate is always of GOOD

alignment and starts with an IN\_COMBAT schedule. The clone is placed in a

spot near to the original NPC.

Parameters:

npc The NPC that should be cloned.

Return:

An object reference to the clone, or zero if the intrinsic did not receive a

valid NPC to clone or if a suitable spot for the clone could not be found.

object UI\_create\_new\_object(int shape)

Creates a new object with the specified shape.

Parameters:

shape The shape in 'SHAPES.VGA' of the new object. If this is a monster

shape, the object is created without any equipment whatsoever, even

if the shape has an associated equipment entry.

Return:

An object reference to the newly created object. This object is pushed into

the 'last created' LIFO stack, and starts out in 'limbo'; it will remain

there until placed either, in the game world through the

'UI\_update\_last\_created' intrinsic or inside a container through the

'UI\_give\_last\_created' intrinsic.

object UI\_create\_new\_object2(int shape, int pos[]) [SI, Exult]

Creates a new object with the specified shape and places it in the game

world at the supplied location.

Parameters:

shape The shape in 'SHAPES.VGA' of the new object. If this is a monster

shape, the object is created with all equipment specified by the

shape's equipment entry.

pos The location where the new object is to be placed.

The location can be specified by arrays ranging from 1 to 4 components. The

meaning of these components is as follows:

\* 1 element: the object will be deleted (equivalent to 'remove\_item'

intrinsic);

\* 2 elements: mean a (x, y) location. The object is placed at z = 0 in the

current game map;

\* 3 elements: mean a (x, y, z) location. The object is placed in the

current game map;

\* 4 elements: mean a (x, y, z, map) location. The object is placed in the

specified map unless it is -1, in which case it will be placed in the

current game map.

Return:

An object reference to the newly created object. This object is placed in

the game world exactly as if UI\_update\_last\_created(pos) had been called,

and hence will \*not\* be in the 'last created' stack.

UI\_create\_barge\_object(int width, int height, int dir) [Exult]

Creates a barge object with the specified width, height and direction.

Parameters:

width The width (x), in tiles, of the barge object.

height The height (y), in tiles, of the barge object.

dir This optional parameter sets the barge's initial direction. If

missing, it is assumed to be north (0).

Return:

An object reference to the newly created barge object. This object is pushed

into the 'last created' LIFO stack, and starts out in 'limbo'; it will

remain there until placed either, in the game world through the

'UI\_update\_last\_created' intrinsic or inside a container through the

'UI\_give\_last\_created' intrinsic.

object UI\_set\_last\_created(object obj)

object obj->set\_last\_created()

Moves the object from the game map or container where it was and places it

in the 'last created' stack. This intrinsic does nothing if the specified

object is already in the aforementioned stack.

Parameters:

obj The object to be moved to the 'last created' stack.

Return:

Returns zero if the object was already in the 'last created' stack, an

object reference to the object otherwise.

bool UI\_update\_last\_created(int pos[])

Pops the last object pushed in the 'last created' stack and places it in the

game map at the supplied position.

Parameters:

pos The location where the new object is to be placed.

The location can be specified by arrays ranging from 1 to 4 components. The

meaning of these components is as follows:

\* 1 element: the object will be deleted (equivalent to 'remove\_item'

intrinsic). The original usecode generally passed '-358' for this

parameter, but this is likely just by convention as any value works.

\* 2 elements: mean a (x, y) location. In Exult, the object is placed at z

= 0 in the current game map; in the original games, nothing is done and

the update fails.

\* 3 elements: mean a (x, y, z) location. The object is placed in the

current game map. In Exult, this always succeeds; it could fail in the

original games.

\* 4 elements: mean a (x, y, z, map) location. Exult-only. The object is

placed in the specified map unless it is -1, in which case it will be

placed in the current game map.

Return:

For the 1-element version, false if the 'last created' stack is empty or

true if the object was deleted. Otherwise, returns true if object was

successfully placed on the map, false otherwise.

\*\*\*\* NOTES ABOUT ORIGINAL GAMES \*\*\*\*

In the original games, the function could fail to place the item in the map.

In this case, it would try to delete an unrelated on-screen object flagged

with the TEMPORARY flag to make room for the updated object and retry. The

retry could fail again, in which case the update fails; otherwise, it would

succeed. This means that, on occasion, using this function could cause

objects to be deleted in the original.

\*\* END NOTES ABOUT ORIGINAL GAMES \*\*

bool UI\_give\_last\_created(object obj)

bool obj->give\_last\_created()

Gives the last item in the 'last created' stack to the specified container

or NPC. This intrinsic ignores volume, and is quite likely to succeed; the

given object is popped from the 'last created' stack only if the intrinsic

succeeds.

Parameters:

obj Container or NPC that will gain the object.

Return:

true if successful, false otherwise.

int[3] UI\_get\_object\_position(object obj)

int[3] obj->get\_object\_position()

Gets the current location the desired object.

Parameters:

obj The object whose position is desired.

Return:

An array containing the (x, y, z) position of the object.

UI\_move\_object(object obj, int pos[], bool flag)

obj->move\_object(int pos[], bool flag)

Moves the specified object to a new location in the game world. This move is

instantaneous, and is often used for teleportation in the original usecode.

Parameters:

obj The object to be moved. Can be PARTY = -357 to teleport the entire

party.

pos The object's new location. Can be a (x, y) position, a (x, y, z)

position or a (x, y, z, map) position. If no z coordinate is

specified, it is assumed to be zero. If no map is specified, or if

it is set to a negative number, the avatar's map will be used

instead.

flag This optional parameter controls controls whether or party members

that are paralyzed or asleep; if true, these members will be

teleported, otherwise, they won't defaults to false.

If the avatar (or the party) is being teleported, the view will center in

the new location and nearby eggs will be activated normally.

int UI\_get\_distance(object obj1, object obj2)

int obj1->get\_distance(obj2)

Gets the distance between two objects.

Parameters:

obj1 One of the two objects.

obj2 The other object.

Return:

The distance between the two objects, or zero if one or both of the

parameters is not a valid object.

int UI\_find\_direction(object param1, object param2)

int param1->find\_direction(object param2)

int UI\_find\_direction(object param1, int param2[3])

int param1->find\_direction(int param2[3])

int UI\_find\_direction(object param1, mixed param2[4])

int param1->find\_direction(mixed param2[4])

int UI\_find\_direction(int param1[3], object param2)

int UI\_find\_direction(int param1[3], int param2[3])

int UI\_find\_direction(int param1[3], mixed param2[4])

int UI\_find\_direction(mixed param1[4], object param2)

int UI\_find\_direction(mixed param1[4], int param2[3])

int UI\_find\_direction(mixed param1[4], mixed param2[4])

Get the direction from one location or object to another.

This intrinsic is aliased under the name 'direction\_from'; they are both

equivalent in Exult.

Parameters:

param1 The object or location \*from\* which we want the direction.

param2 The object or location \*to\* which we want the direction.

Both parameters can be objects, arrays with 3 components or arrays with 4

components. Depending on what they are, Exult will treat them differently

and expect them to conform to a certain format; specifically:

Type Interpretation/Format

object The parameter is an object or NPC.

int [3] The parameter is a (x, y, z) position, such as the return

of a UI\_get\_object\_position intrinsic call.

mixed [4] The parameter is of the form (object, x, y, z), such as the

return of a call to UI\_click\_on\_item intrinsic.

Return:

The 8-cardinal-point direction of an arrow starting in the first parameter

and pointing to the second parameter. It can be one of the following values:

NORTH = 0 NORTHEAST = 1 EAST = 2 SOUTHEAST = 3

SOUTH = 4 SOUTHWEST = 5 WEST = 6 NORTHWEST = 7

int UI\_add\_cont\_items(object obj, int count, int shape, int [SI, Exult]

qual, int frame, bool temp)

int obj->add\_cont\_items(int count, int shape, int qual, int frame, bool temp)

Create a number of objects inside a container.

Parameters:

obj The container or NPC that will receive the created objects.

count The amount of objects to be created.

shape The shape in 'SHAPES.VGA' that should be used for the created

objects.

qual The quality of the objects being created. Can be QUALITY\_ANY = -359

to create items of any quality.

frame What frame will be used for the created objects. Can be FRAME\_ANY =

-359 to create items of any frame.

temp Flag indicating whether the created items are temporary or not.

Return:

How many objects were actually added to the container.

int UI\_remove\_cont\_items(object obj, int count, int shape, [SI, Exult]

int qual, int frame, bool unk)

int obj->remove\_cont\_items(int count, int shape, int qual, int frame, bool unk)

Removes 'count' objects with shape 'shape', quality 'qual' and frame 'frame'

that are contained by 'obj'. 'unk' is an unknown flag which is ignored by

Exult; I \*think\* it is safe to omit it from the parameter list, but I am not

sure. 'qual' and 'frame' can be wildcards:

qual frame

QUALITY\_ANY = -359 FRAME\_ANY = -359

Return value is the number of objects actually removed.

Looks through the contents of a container and deletes a specified amount of

objects matching a given search criteria.

Parameters:

obj The object whose contents are to be searched. Can be a container,

an NPC or PARTY = -357 to search the whole party.

count The amount of objects to be deleted. Can be QUANTITY\_ANY = -359 to

delete all objects that match the criteria.

shape The shape we are looking for.

qual The quality that is being sought. Can be QUALITY\_ANY = -359 to

match any quality.

frame What frame is desired. Can be FRAME\_ANY = -359 to match any frame.

unk Flag of unknown meaning. This parameter is ignored by Exult, and

can be omitted.

Return:

The number of objects actually removed.

UI\_remove\_from\_area(int shape, int frame, int ul[2], int [SI, Exult]

lr[2])

Scans a rectangular area for objects matching the given criteria and deletes

all found objects.

Parameters:

shape The shape in 'SHAPES.VGA' that is to be searched.

frame The frame being sought.

ul A (x, y) position specifying the upper-left corner of the region to

be searched.

lr A (x, y) position specifying the lower-right corner of the region

to be searched

================================================================================

OBJECT INTRINSICS

================================================================================

These intrinsics are used to get or set properties of an object, such as

quantity, quality or current frame.

int UI\_get\_item\_shape(object obj)

int obj->get\_item\_shape()

Gets the current shape of the desired object.

Parameters:

obj The object whose shape is desired.

Return:

The shape in 'SHAPES.VGA' of the object in question.

UI\_set\_item\_shape(object obj, int shape)

obj->set\_item\_shape(int shape)

Changes the shape of an object.

Parameters:

obj The object whose shape is to be changed.

shape The shape in 'SHAPES.VGA' which the object will assume.

int UI\_get\_item\_frame(object obj)

int obj->get\_item\_frame()

Gets the current frame of the desired object, \*without\* the rotation bit.

Parameters:

obj The object whose frame is desired.

Return:

The frame from 'SHAPES.VGA' of the object in question. Only the base frame

is returned, without any rotation information.

UI\_set\_item\_frame(object obj, int frame)

obj->set\_item\_frame(int frame)

Sets the current frame of the desired object, preserving the rotation bit.

Parameters:

obj The object whose frame is to be changed.

frame The desired value for the frame.

int UI\_get\_item\_frame\_rot(object obj)

int obj->get\_item\_frame\_rot()

Gets the current frame of the desired object, \*with\* the rotation bit.

Parameters:

obj The object whose frame is desired.

Return:

The frame from 'SHAPES.VGA' of the object in question; including the

rotation bit.

UI\_set\_item\_frame\_rot(object obj, int frame)

obj->set\_item\_frame\_rot(int frame)

Sets the current frame of the desired object; does \*not\* preserve the

rotation bit.

Parameters:

obj The object whose frame is to be changed.

frame The desired value for the frame.

int UI\_get\_item\_quality(object obj)

int obj->get\_item\_quality()

Gets the quality of the desired object.

Parameters:

obj The object whose quality is desired.

Return:

The quality of the object if it has it, zero otherwise.

bool UI\_set\_item\_quality(object obj, int qual)

bool obj->set\_item\_quality(int qual)

Sets the quality of the desired object.

Parameters:

obj The object whose quality is to be changed.

qual The desired value for the quality. Quality is unchanged if qual ==

QUALITY\_ANY.

Return:

Returns true if the object has a quality or if qual == QUALITY\_ANY, false

otherwise.

int UI\_get\_item\_quantity(object obj)

int obj->get\_item\_quantity()

int UI\_get\_item\_quantity(object obj, int unk)

int obj->get\_item\_quantity(int unk)

Gets the quantity of the desired object.

Parameters:

obj The object whose quantity is desired.

unk An unknown parameter from the original games. Can be omitted if

desired, as Exult does not use it.

Return:

The quantity of the object if it has it, '1' otherwise.

bool UI\_set\_item\_quantity(object obj, int quant)

bool obj->set\_item\_quantity(int quant)

Sets the quantity of the desired object. The object's quantity is changed

only if the object has it.

Parameters:

obj The object whose quantity is to be changed.

quant The desired value for the quantity. If this value is zero, the

object will be deleted unless it is not in the game map -- likely

because it has been created by Usecode but not yet placed.

Return:

Returns true if the object has a quantity, false otherwise.

int UI\_get\_lift(object obj)

int obj->get\_lift()

Gets how far above the ground the object is.

Parameters:

obj The object whose lift is desired.

Return:

The object's z coordinate.

UI\_set\_lift(object obj, int lift)

obj->set\_lift(int lift)

Moves the object on the vertical direction to a desired height off the

ground.

Parameters:

obj The object whose lift is to be set.

lift How far above the ground the object is going to be. Must be in the

range 0 to 20, but heights above 15 will not be saved (as of

2009/02/26).

int UI\_get\_item\_flag(object obj, int flag)

int obj->get\_item\_flag(int flag)

Inspects the chosen flag of the given object.

Parameters:

obj The object whose flag we wish to inspect. Not all flags are

available for all objects.

flag What we want to know. This is a number in the 0 to 63 range.

The flag parameter is a numeric value describing a (usually boolean) bit of

information about the object in question. Valid pre-defined values for the

flag are:

INVISIBLE = 0 Object is invisible (rendered with

'invisible' palette, or not at all).

ASLEEP = 1 Object is asleep. Only really useful for

NPCs.

CHARMED = 2 Object is charmed (attacks allies). Only

really useful for NPCs.

CURSED = 3 Object is cursed (stats malus). Only really

useful for NPCs.

DEAD = 4 Object is dead. Only really useful for

NPC/monster. \*DONT set on dead bodies, causes

bugs.\*

IN\_PARTY = 6 Object is an NPC in the party.

PARALYZED = 7 Object is paralyzed and unable to move.

Normal objects will be stationary by default.

PARALYSED = 7 Canadian/British spelling.

POISONED = 8 Object is afflicted with poison. Can be set

on objects, but will only do anything to

NPCs.

PROTECTION = 9 Object is protected from harm. Only really

useful for NPCs.

ON\_MOVING\_BARGE = 10 Object is on a 'barge' object that is moving.

OKAY\_TO\_TAKE = 11 Object will not trigger stealing usecode if

you take it.

MIGHT = 12 Object has might (Stats boon). Only really

useful for NPCs.

IMMUNITIES = 13 Object has several immunities. Can be

inspected only.

CANT\_DIE = 14 Object can't die. Used for testing purposes,

and certain critical NPCs (such as L.B.). Can

be inspected only.

IN\_ACTION = 15 Object is executing the usecode called by the

Dance spell.

DONT\_MOVE = 16 SI: The NPC cannot move.

DONT\_RENDER = 16 BG: The NPC cannot move, and is not rendered

at all.

SI\_ON\_MOVING\_BARGE = 17 SI: Object is on a special instance of the SI

barges. (Turtle for example.)

TEMPORARY = 18 Object is temporary and will decay once the

chunk it is in is un-cached.

ACTIVE\_SAILOR = 20 Object is the 'captain' of a barge. When

getting the flag, you will actually get the

current captain.

OKAY\_TO\_LAND = 21 Set in usecode for flying carpet, TRUE if you

can land currently.

BG\_DONT\_MOVE = 22 BG: BG's version of DONT\_MOVE.

SI\_DONT\_RENDER = 22 SI: SI's version of DONT\_RENDER.

IN\_DUNGEON = 23 If set on PC or an NPC, they won't trigger

the stealing usecode if they take food

objects.

IS\_SOLID = 24 Used by gangplank usecode to determine if an

object is solid.

CONFUSED = 25 This will be set if the avatar failed

copyright protection questions in SI. Only

really meaningful used on the avatar.

ACTIVE\_BARGE = 26 Object is a barge object moving, or on a

barge object that is moving. Set in usecode,

and mostly used for the SI 'NPC' ships such

as the turtle.

MET = 28 Object has been talked to previously. Should

be set in a conversation Usecode script. This

determines conversation behavior, and whether

the NPC's real name or shape name is

displayed when they are single-clicked on. BG

originally used global flags for this, which

amounts to an extra 250-odd flags. What a

waste of time.

SI\_TOURNAMENT = 29 Literal plot immunity: the NPC can die only

if the plot allows it. If enough damage is

taken in a single blow, the object will call

usecode (eventid = 7) on death. Used for the

List Field in SI.

SI\_ZOMBIE = 30 Object will not respond to normal cues. Used

for sick Neyobi, insane party members/Cantra.

NO\_SPELL\_CASTING = 31 Object cannot cast spells. Set when the

magebane strikes the object.

POLYMORPH = 32 Object is polymorphed.

TATTOOED = 33 Object has been tattooed. Set after the lady

tattoos you by usecode. Causes the avatar

portraits to use a different 'tattooed' shape

num.

READ = 34 Object can read serpent- or rune-script text.

Only really meaningful used on the avatar.

PETRA = 35 Object has switched bodies with Petra. This

changes the portraits of NPCs according to a

list in the 'avatar\_data.txt' file. Only

really meaningful for NPCs in that list.

FLIGHT = 36 Object can fly. Only really useful for NPCs.

FREEZE = 37 Object is freezing. Only really useful for

NPCs.

NAKED = 38 Object is naked. On the avatar, causes

him/her to use the appropriate 'naked' shape.

On other NPCs there no effect. To actually

make NPCs naked, use set\_polymorph instead.

Return:

For most flags, true if the flag is set (that is, in effect), false

otherwise. There is one exception, however: The 'ACTIVE\_SAILOR' and

'ACTIVE\_BARGE' flags have a peculiar behavior in that they actually returns

an object reference to the last object to have that flag set, or zero if no

object has been set, or if the flag has been cleared with 'clear\_item\_flag'

intrinsic for the given flag.

With the exception of the OKAY\_TO\_TAKE, TEMPORARY and INVISIBLE flags, these

flags are valid for NPCs only. You can get, set or clear any of those values

and Exult will not complain; however, it will also not save the values of

the flags except for the three mentioned before. Moreover, the INVISIBLE

flag will not be saved for objects unless they are of 'quality flags' class.

UI\_set\_item\_flag(object obj, int flag)

obj->set\_item\_flag(int flag)

Sets the desired flag for the given object. Many flags have immediate

effects when set; most do not. See the get\_item\_flag intrinsic for flag

values.

Parameters:

obj The object whose flag we wish to set. Not all flags are available

for all objects.

flag What flag we want to set. This is a number in the 0 to 63 range; it

can be any of the values for the 'flag' parameter of the

set\_item\_flag intrinsic.

In the special case of the 'ACTIVE\_SAILOR' and 'ACTIVE\_BARGE' flags, stores

the supplied object, instead of a flag value. This stored object will be

returned by future calls of the 'get\_item\_flag' intrinsic for the given

flag.

With the exception of the OKAY\_TO\_TAKE, TEMPORARY and INVISIBLE flags, these

flags are valid for NPCs only. You can get, set or clear any of those values

and Exult will not complain; however, it will also not save the values of

the flags except for the three mentioned before. Moreover, the INVISIBLE

flag will not be saved for objects unless they are of 'quality flags' class.

UI\_clear\_item\_flag(object obj, int flag)

obj->clear\_item\_flag(int flag)

Clears the desired flag for the given object. Many flags have immediate

effects when cleared; most do not. See the get\_item\_flag intrinsic for flag

values.

Parameters:

obj The object whose flag we wish to clear. Not all flags are available

for all objects.

flag What flag we want to clear. This is a number in the 0 to 63 range;

it can be any of the values for the 'flag' parameter of the

set\_item\_flag intrinsic.

In the special case of the 'ACTIVE\_SAILOR' and 'ACTIVE\_BARGE' flags, the

object that they stores is set to zero. This will be the return of any

future calls to the 'get\_item\_flag' intrinsic for the given flag.

With the exception of the OKAY\_TO\_TAKE, TEMPORARY and INVISIBLE flags, these

flags are valid for NPCs only. You can get, set or clear any of those values

and Exult will not complain; however, it will also not save the values of

the flags except for the three mentioned before. Moreover, the INVISIBLE

flag will not be saved for objects unless they are of 'quality flags' class.

object UI\_get\_barge(object obj)

object obj->get\_barge()

Checks the barge upon which the supplied object is on.

Parameters:

obj The object to be inspected.

Return:

Returns zero if no valid object is passed, or if the object is not lying on

a barge. Otherwise, returns the object reference of the barge (shape 961)

object upon which the supplied object lies.

UI\_set\_barge\_dir(object obj, int dir) [SI, Exult]

obj->set\_barge\_dir(int dir)

If 'obj' is a barge object, sets its direction (along with all objects

laying on 'obj') to 'dir'. 'dir' can be one of the constants defined for

'find\_direction' intrinsic.

Parameters:

obj The object to be animated.

dir The 8-cardinal-point direction that the barge should face. This can

be one of the values returned by the find\_direction intrinsic.

bool UI\_on\_barge()

Determines if the entire party is inside the same barge as the avatar.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In SI, this intrinsic also forces the barge to refresh the list of all

objects lying on it.

\*\*\*\* END SI SPECIFIC \*\*\*\*

Return:

If the avatar is on a barge and all party members are on that same barge,

returns true; in all other cases, returns false.

object UI\_get\_container(object obj)

object obj->get\_container()

Gets the container of the supplied object.

Parameters:

obj The object whose container is desired.

Return:

Return is zero if the object is directly in the game world, or an object

reference to its immediate container otherwise.

UI\_remove\_item(object obj)

obj->remove\_item()

Removes the object from the game world and from the 'last created' stack. If

the object is not an NPC, it will de deleted; NPCs will still exist.

Parameters:

obj The object to be deleted.

int UI\_get\_usecode\_fun(object obj) [Exult]

int obj->get\_usecode\_fun()

Gets the usecode function number of an object. This takes into consideration

a NPC's assigned function (if any) and the correct shape <> function mapping

for shapes > 1024.

Parameters:

obj The object whose usecode function we desire.

Return:

The usecode function number that is assigned to the object.

UI\_set\_usecode\_fun(actor npc, function fun) [Exult]

npc->set\_usecode\_fun(function fun)

Assigns a new usecode function to an NPC.

Parameters:

npc The NPC whose usecode function will be changed.

fun The new usecode function.

int UI\_get\_map\_num(object obj) [Exult]

int obj->get\_map\_num()

Checks which map an object is in.

Parameters:

obj The object whose map is desired.

Return:

The map in which the object is located, or -1 if the object is nowhere.

int UI\_get\_item\_weight(object obj) [SI, Exult]

int obj->get\_item\_weight()

Checks an object's weight.

Parameters:

obj The object whose weight is desired.

Return:

The object's weight.

bool UI\_is\_on\_keyring(int qual) [SI]

Checks if a key is in the keyring. Note that the keyring is a global object;

multiple keyrings will share the same list of keys.

Parameters:

qual The quality of the key to be checked.

Return:

true if the key is on the keyring, false otherwise.

UI\_add\_to\_keyring(int qual) [SI]

Adds a key to the keyring. Note that the keyring is a global object;

multiple keyrings will share the same list of keys.

Parameters:

qual The quality of the key to be added.

bool UI\_remove\_from\_keyring(int qual) [Exult: SI]

Removes a key from the keyring. Note that the keyring is a global object;

multiple keyrings will share the same list of keys.

Parameters:

qual The quality of the key to be removed.

Return:

If the key was on the keyring, returns true; otherwise, returns false.

int UI\_apply\_damage(int base, int hits, int type, object obj)

Causes random damage of a given type to the desired object, subject to

armor.

Parameters:

base The 'strength' of the attack; only one third of this value (rounded

down) is used to deal damage. This parameter is ignored depending on

damage type; specifically, damage of LIGHTNING\_DAMAGE type ignores

this parameter.

hits The 'weapon points' of the attack. If this parameter is equal to 127

or more, this intrinsic will ignore all armor and deal a fixed 127

points of damage of the specified type, ignoring the 'base'

parameter.

type The type of the damage. If the target is immune to this damage type,

he will take no damage from this intrinsic; conversely, a monster

vulnerable to this type of damage takes doubled damage.

Armor-granted immunities also protect the target from the effects of

this intrinsic. If the damage type is LIGHTNING\_DAMAGE,

ETHEREAL\_DAMAGE or SONIC\_DAMAGE, this intrinsic will ignore armor.

obj The object to be damaged.

The type of the damage is a numerical value that describes what kind of

damage should be caused. It can be one of the following values:

NORMAL\_DAMAGE = 0 Normal damage.

FIRE\_DAMAGE = 1 Damage from heat sources.

MAGIC\_DAMAGE = 2 Damage from a magical source.

LIGHTNING\_DAMAGE = 3 Damage from electrical sources such as

lightning. Extremely painful, even a single

point of damage causes the screen to flash a

red color. This damage ignores all armor and

does not depend on the attacker's strength.

POISON\_DAMAGE = 3 Damage from poison. Extremely painful, even a

single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

STARVATION\_DAMAGE = 3 Damage from starvation. Extremely painful, even

a single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

FREEZING\_DAMAGE = 3 Damage from frostbite. Extremely painful, even

a single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

ETHEREAL\_DAMAGE = 4 Special magical damage, basically magic damage

not blocked by normal magic resistance or

armor.

SONIC\_DAMAGE = 5 Sound-based damage. This damage type ignores

armor.

The damage is calculated from a series of generated random numbers. It

proceeds along the following steps:

\* If the target is wearing any armor that makes him/her immune to the

specified damage type, he takes no damage from this intrinsic.

\* If hits is 127 or more, 127 points of damage are caused to the

target. This case ignores all armor.

\* If base/3 (rounded down) is zero, it does not affect damage;

likewise, if hits is zero, it does not affect damage. A random number

will be generated for each of them that are nonzero, ranging from 1 to

base/3 or from 1 to hits, as appropriate. These two random numbers are

added together.

\* If the damage type does not ignore armor, the armor value of

everything worn by the target is computed. Monster-specified armor

values are also added to this value. If the total armor is at least

zero, a random number in the range 1 to (total armor) is generated and

subtracted from the damage total above.

\* If the target has immunities granted by his monster data, he will

take no damage; if he has any vulnerabilities, the remaining damage,

if positive, will be doubled.

Return:

Returns true if any damage is caused on a valid target, false otherwise.

UI\_reduce\_health(object obj, int hits, int type)

obj->reduce\_health(int hits, int type)

Causes a fixed amount of damage of the specified type to a target. This

intrinsic ignores armor and armor-granted immunities.

Parameters:

obj The object to be damaged.

hits How much damage is to be caused.

type The type of the damage. If the target is immune to this damage type,

he will take no damage from this intrinsic; conversely, a monster

vulnerable to this type of damage takes doubled damage.

Armor-granted immunities are ignored.

The type of the damage is a numerical value that describes what kind of

damage should be caused. It can be one of the following values:

NORMAL\_DAMAGE = 0 Normal damage.

FIRE\_DAMAGE = 1 Damage from heat sources.

MAGIC\_DAMAGE = 2 Damage from a magical source.

LIGHTNING\_DAMAGE = 3 Damage from electrical sources such as

lightning. Extremely painful, even a single

point of damage causes the screen to flash a

red color. This damage ignores all armor and

does not depend on the attacker's strength.

POISON\_DAMAGE = 3 Damage from poison. Extremely painful, even a

single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

STARVATION\_DAMAGE = 3 Damage from starvation. Extremely painful, even

a single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

FREEZING\_DAMAGE = 3 Damage from frostbite. Extremely painful, even

a single point of damage causes the screen to

flash a red color. This damage ignores all

armor and does not depend on the attacker's

strength.

ETHEREAL\_DAMAGE = 4 Special magical damage, basically magic damage

not blocked by normal magic resistance or

armor.

SONIC\_DAMAGE = 5 Sound-based damage. This damage type ignores

armor.

int UI\_attack\_object(object attacker, object target, int weapon)

int attacker->attack\_object(object target, int weapon)

Causes a NPC to attack the object with a given weapon. This intrinsic does

not animate the NPC in any way; it only emulates a hit with the selected

weapon.

Parameters:

attacker Who is attacking.

target What is being attacked. Must be an object or NPC.

weapon What shape in 'SHAPES.VGA' to use as the source of the weapon

information in the attack, or -1 for an unarmed attack. If this

is a missile or thrown weapon, it will consume the appropriate

ammunition (if any) and fire the correct projectile.

If the NPC is targeting a tile, the attack will only be meaningful if the

weapon causes an explosion.

Return:

true if a melee attack hits or if a ranged attack fires its projectile;

returns false if the melee attack missed or if the attack could not be

realized for any reason (including an invalid attacker, an invalid target,

being out of range, lack of adequate ammunition for the given weapon) if

attacking a valid target with a valid weapon, false otherwise.

UI\_fire\_projectile(object obj, int dir, int proj, int attpts, int weapon, int

ammo)

obj->fire\_projectile(int dir, int proj, int attpts, int weapon, int ammo)

Causes the given object to fire a high-speed missile in the supplied

direction.

Parameters:

obj The object that will fire the projectile.

dir Numeric value indicating the direction in which the projectile

will be fired.

proj The shape in 'SHAPES.VGA' of the projectile to be fired.

attpts Numeric value determining the likelihood that the missile will

hit. Works like the 'att' parameter in the roll\_to\_win intrinsic

in a contest against the target's COMBAT property.

weapon The shape in 'SHAPES.VGA' whose weapon information will be used

when the missile hits.

ammo The shape in 'SHAPES.VGA' whose ammunition information will be

used when the missile hits.

The direction into which the projectile is filed is the 8-cardinal-point

direction in which the projectile will travel. It can be one of the

following values:

NORTH = 0 NORTHEAST = 1 EAST = 2 SOUTHEAST = 3

SOUTH = 4 SOUTHWEST = 5 WEST = 6 NORTHWEST = 7

================================================================================

NPC INTRINSICS

================================================================================

These intrinsics allow you to get or set NPC-specific properties of a NPC,

such as schedules, alignment or stats.

int UI\_get\_npc\_object(int npc)

int npc->get\_npc\_object()

int[] UI\_get\_npc\_object(int npc[])

Gets the NPC object references of one or more NPC numbers.

Parameters:

npc The NPC number, or array of NPC numbers, whose object references are

desired.

Return:

If the input is an array of NPC numbers, the return is an array of NPC

object references. The output array places the NPC references in the same

order they are in the input array.

Otherwise, the return is the object reference of the given NPC number.

int UI\_get\_schedule\_type(actor npc)

int npc->get\_schedule\_type()

Gets the current schedule of an NPC.

Parameters:

npc The NPC whose schedule is desired.

Return:

Zero if 'npc' is not a valid NPC; otherwise, an integer associated to the

NPC's current schedule. This is the same value that is reported in the cheat

screen, and can be one of the following values:

IN\_COMBAT = 0 PACE\_HORIZONTAL = 1 PACE\_VERTICAL = 2

TALK = 3 DANCE = 4 EAT = 5

FARM = 6 TEND\_SHOP = 7 MINE = 8

MINER = 8 HOUND = 9 STANDTHERE = 10

LOITER = 11 WANDER = 12 BLACKSMITH = 13

SLEEP = 14 WAIT = 15 MAJOR\_SIT = 16

GRAZE = 17 BAKE = 18 SEW = 19

SHY = 20 LAB = 21 THIEF = 22

WAITER = 23 SPECIAL = 24 KID\_GAMES = 25

TAG = 25 EAT\_AT\_INN = 26 DUEL = 27

SPAR = 27 PREACH = 28 PATROL = 29

DESK\_WORK = 30 FOLLOW\_AVATAR = 31

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

If the NPC is traversing a path created by a call to either,

UI\_path\_run\_usecode or UI\_si\_path\_run\_usecode intrinsics, this intrinsic

will return the value WALK\_TO\_SCHEDULE = 32 instead of the values above.

\*\*\*\* END SI SPECIFIC \*\*\*\*

UI\_set\_schedule\_type(actor npc, int sched)

npc->set\_schedule\_type(int sched)

Gets the current schedule of an NPC.

Parameters:

npc The NPC whose schedule we wish to change.

sched The new schedule type. Can be any of the values listed as the

return of a get\_schedule\_type intrinsic call.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

The SI-specific value WALK\_TO\_SCHEDULE = 32 \*cannot\* be set with this

intrinsic; only a call to either, UI\_path\_run\_usecode or

UI\_si\_path\_run\_usecode intrinsics can do that.

\*\*\*\* END SI SPECIFIC \*\*\*\*

UI\_modify\_schedule(actor npc, int time, int sched, int [SI, Exult]

pos[2])

npc->modify\_schedule(int time, int sched, int pos[2])

Changes the schedule table of an NPC for the desired time period. This new

schedule is persistent across save games. The NPC will execute his/her new

schedule in the map he/she currently resides when the time comes.

Parameters:

npc The NPC whose schedule table is to be changed.

time An integer specifying the time period of the new schedule. Must be

an integer in the 0-7 range, such as those returned by the

part\_of\_day intrinsic.

sched An integer specifying the new schedule. Must be an integer in the

0-31 range, such as those returned by the get\_schedule\_type

intrinsic.

loc A (x, y) location, specifying the location of the new schedule.

UI\_set\_new\_schedules(actor npc, int time, int sched, int [SI, Exult]

loc[2])

UI\_set\_new\_schedules(actor npc, int time[n], int sched[n], int loc[2n])

npc->set\_new\_schedules(int time, int sched, int loc[2])

npc->set\_new\_schedules(int time[n], int sched[n], int loc[2n])

Changes the schedule table of an NPC. This new schedule table is persistent

across save games. The NPC will execute his/her new schedules in the map

he/she currently resides.

Parameters:

npc The NPC whose schedule table is to be changed.

time An integer, or array of integers, specifying the time periods of

the new schedules. Each element of the array is an integer in the

0-7 range, such as those returned by the part\_of\_day intrinsic.

sched An integer, or array of schedules, specifying the new schedules.

There must be one such schedule for each time period supplied in

the 'time' parameter. Each element of this array is an integer in

the 0-31 range, such as those returned by the get\_schedule\_type

intrinsic.

loc An array of (x, y) positions, specifying the new schedule

locations. This array must have a (x, y) location for each time

period supplied in the 'time' parameter.

UI\_run\_schedule(actor npc) [SI, Exult]

npc->run\_schedule()

Forces an NPC to return to his/her regularly scheduled activities.

Parameters:

npc The NPC that should be minding his own business.

UI\_revert\_schedule(actor npc) [SI, Exult]

npc->revert\_schedule()

Reverts the schedules of a NPC to those stored in 'STATIC/schedule.dat'. Can

be risky for NPCs not in the original games.

Parameters:

npc The NPC whose schedule is to be reset.

int UI\_get\_npc\_prop(object obj, int prop)

int UI\_get\_npc\_prop(object obj, string prop)

int obj->get\_npc\_prop(int prop)

int obj->get\_npc\_prop(string prop)

Gets the current value of an object's or NPC's property. Despite the name,

this intrinsic can also be called for any object, although most properties

will be unavailable for anything but NPCs.

Parameters:

obj The object or NPC whose property is desired.

prop String or integer specifying the property to be inspected.

If the desired property has been specified as an integer, it must be in the

range from 0 to 11. It can be one of the following values:

STRENGTH = 0 DEXTERITY = 1 INTELLIGENCE = 2

HEALTH = 3 COMBAT = 4 MANA = 5

MAX\_MANA = 6 TRAINING = 7 EXPERIENCE = 8

FOODLEVEL = 9 SEX\_FLAG = 10 MISSILE\_WEAPON = 11

Return:

Returns the current value of the specified property. The following

properties are special cases:

HEALTH This is the only meaningful property for non-NPC objects.

Returns the current remaining hit points of the object or

NPC.

SEX\_FLAG The return is true if the NPC is female, false otherwise.

MISSILE\_WEAPON The return is true if the NPC is using a missile weapon or

a good thrown weapon (such as a spear), false otherwise.

This intrinsic returns zero for any property which is not meaningful for the

supplied object; this includes string properties that have not been

previously set with a UI\_set\_npc\_prop intrinsic call.

\*\*\*\*\*\* BG SPECIFIC \*\*\*\*\*\*

Standard BG does not have sex data for anyone but the avatar, so the return

is almost always 'false'. This is not a limitation of the engine or of

Exult, just a statement about the game data.

\*\*\*\* END BG SPECIFIC \*\*\*\*

UI\_set\_npc\_prop(object obj, int prop, int delta)

UI\_set\_npc\_prop(object obj, string prop, int delta)

obj->set\_npc\_prop(int prop, int delta)

obj->set\_npc\_prop(string prop, int delta)

Changes an object's or NPC's property by the specified value. Despite the

name, this intrinsic can also be called for any object, although most

properties will be unavailable for anything but NPCs.

Parameters:

obj The object or NPC whose property is to be changed.

prop String or integer specifying the property to be changed.

delta The desired numeric \*change\* in the specified property.

If the desired property has been specified as an integer, it must be one of

the values listed in the UI\_get\_npc\_prop intrinsic description. The

following values are special cases:

HEALTH This is the only meaningful property for non-NPC objects.

Changes the remaining hit points of the object or NPC. If

the remaining hit points become zero, an object will

become \*indestructible\*.

EXPERIENCE This property is changed by /half/ of the specified value.

SEX\_FLAG A nonzero 'delta' sets the NPC to female, while a zero

'delta' sets the NPC to male.

MISSILE\_WEAPON Nothing happens when this property value is supplied.

\*\*\*\*\*\* BG SPECIFIC \*\*\*\*\*\*

In the original BG, you could not set the sex flag except by hex-editing the

game data. Exult does not suffer from this limitation.

\*\*\*\* END BG SPECIFIC \*\*\*\*

If the desired property has been specified as a string, it will be created

if it does not exist; its initial value is assumed to be zero in this case.

Only NPCs can have such properties, and they will be displayed in the NPC's

stats window.

Return:

Always returns '0'.

string UI\_get\_npc\_name(object obj)

string obj->get\_npc\_name()

string[] UI\_get\_npc\_name(object obj[])

Gets the name of one or more objects. Despite the name, this intrinsic can

be called for any object, not just NPCs; for non-NPCs, the return is the

text displayed when you click the object.

Parameters:

obj The object, or array of objects, whose names are desired.

Return:

If the input is an array of objects, the return is an array with the names

of these objects. The output array places the object names in the same order

as the objects are in the input array.

Otherwise, the return is the name of the given object.

UI\_set\_npc\_name(actor npc, string name)

npc->set\_npc\_name(string name)

Changes the name of an NPC.

Parameters:

npc The NPC whose name is to be changed.

name The new name.

UI\_is\_npc(mixed obj)

obj->is\_npc()

Determines if a given object is an NPC.

Parameters:

obj The object to be checked.

Return:

true if the object is an NPC, false otherwise.

bool UI\_is\_dead(actor npc)

bool npc->is\_dead()

Checks if the NPC is dead or alive. This is equivalent to checking if the

'DEAD' flag is set.

Parameters:

npc Who we want to check.

Return:

Returns true if the NPC is dead, false otherwise.

int UI\_get\_npc\_number(actor npc)

int npc->get\_npc\_number()

Parameters:

npc The NPC whose number is desired. This is intended to receive an

object reference, but accepts integers as normal.

Return:

The NPC's number. This is a negative number for all 'true' NPCs, -356 for

the avatar, 1 for monsters or zero for anything that is not a valid NPC.

int UI\_get\_alignment(actor npc)

int npc->get\_alignment()

Gets the NPC's attitude towards the avatar.

Parameters:

npc The NPC whose alignment is desired.

Return:

Returns a number describing the NPC's attitude towards the avatar. Can be

one of the following values:

NEUTRAL = 0 GOOD = 1 EVIL = 2

CHAOTIC = 3

UI\_set\_alignment(actor npc, int align)

npc->set\_alignment(int align)

Sets an NPC's attitude towards the avatar. This change forces the NPC to

reevaluate his targets, and may stop the NPC from fleeing.

Parameters:

npc The NPC to be adjusted.

align The new attitude towards the avatar. Can be any of the return

values of get\_alignment intrinsic.

UI\_remove\_npc(actor npc)

npc->remove\_npc()

Removes the NPC from the game world. The NPC is effectively placed in

'limbo', and has its schedule set to 'WAIT'.

Parameters:

npc The NPC to be removed.

UI\_kill\_npc(actor npc)

npc->kill\_npc()

Kills the specified victim.

Parameters:

npc The soon-to-be-dead NPC.

int UI\_get\_attack\_mode(actor npc) [SI, Exult]

int npc->get\_attack\_mode()

Gets the NPC's attack mode.

Parameters:

npc The NPC whose attack mode is desired.

Return:

A numeric value specifying how the NPC behaves in combat. Possible values

are:

NEAREST = 0 WEAKEST = 1 STRONGEST = 2 BERSERK = 3

PROTECT = 4 DEFEND = 5 FLANK = 6 FLEE = 7

RANDOM = 8 MANUAL = 9

UI\_set\_attack\_mode(actor npc, int mode)

npc->set\_attack\_mode(int mode)

Changes the NPC's attack mode to the desired value.

Parameters:

npc The NPC whose attack mode is to be changed.

mode Numeric value determining how the NPC should behave in combat. Can

be one of the return values of get\_attack\_mode intrinsic.

int UI\_get\_npc\_id(actor npc) [SI, Exult]

int npc->get\_npc\_id()

Retrieves a NPC's identity. This identity has no real effect other than that

given to it by usecode.

Parameters:

npc The NPC whose identity is sought.

Return:

If given a valid NPC, this intrinsic returns that NPC's identity; otherwise

it returns zero.

UI\_set\_npc\_id(actor npc, int id) [SI, Exult]

npc->set\_npc\_id(int id)

Changes an NPC's identity. This identity has no real effect other than that

given to it by usecode.

Parameters:

npc The NPC whose identity is to be changed.

id The new identity value. Must be an integer in the 0-31 range; values

outside this range are accepted but incorrectly saved.

actor UI\_get\_oppressor(actor npc) [SI, Exult]

actor npc->get\_oppressor()

Gets whoever is attacking the supplied NPC. This oppressor can be set

through usecode, but it is also automatically set to whomever has last

attacked (caused damage to) the supplied NPC. It is also set whenever an NPC

acquires a new target -- this NPC will be set as its target's oppressor.

Parameters:

npc The NPC whose oppressor is desired.

Return:

Returns the NPC number of the supplied NPC's oppressor, or zero if no valid

NPC was supplied. The return value has the opposite sign from what you get

from a get\_npc\_number intrinsic call.

UI\_set\_oppressor(actor npc, actor opp) [SI, Exult]

npc->set\_oppressor(opp)

Sets the supplied NPCs oppressor to another NPC.

Parameters:

npc The NPC whose oppressor is to be changed.

opp The NPC that is to be the new oppressor, which must be distinct from

the NPC we are altering. Must be a true NPC; an egg- or usecode-

spawned monster has no real NPC number in Exult, and cannot be an

oppressor.

object UI\_get\_readied(actor npc, int spot) [SI, Exult]

object npc->get\_readied(int spot)

Inspects the given equipment slot in an NPC.

Parameters:

npc The NPC we wish to inspect.

spot What equipment slot we wish to inspect. Can be any of the values of

'spot' for is\_readied intrinsic.

Return:

Returns zero for an invalid spot or if the spot is empty, otherwise returns

an object reference to the object that resides in the desired spot.

bool UI\_is\_readied(actor npc, int spot, int shape, int frame)

npc->is\_readied(int spot, int shape, int frame)

Returns 'true' if an object with shape 'shape' and frame 'frame' is readied

in ready spot 'spot' of 'npc', 'false' otherwise. 'frame' can be the

wildcard 'FRAME\_ANY'. Valid values for 'spot' are:

For SI, it is better to use 'get\_readied' intrinsic instead.

Checks to see if the given NPC is wearing a piece of equipment on a

specified location.

Parameters:

npc The NPC whose equipment is to be inspected.

spot The spot we are checking.

shape The shape we are looking for.

frame The frame we are looking for. Can be FRAME\_ANY = -359 to match any

frame.

\*\*\*\*\*\* BG SPECIFIC \*\*\*\*\*\*

In BG, the 'spot' parameter is limited to the 0-12 range. It can take any of

the following values:

BG\_BACKPACK = 0 Containers worn on the back, such as backpacks

BG\_WEAPON\_HAND = 1 Item wielded in weapon hand

BG\_SHIELD\_HAND = 2 Off-hand is also called the shield hand

BG\_OFF\_HAND = 2 Item wielded in off-hand

BG\_BELT = 3 Items worn around the waist such as belts, and

girdles

BG\_NECK = 4 Items worn around the neck such as amulets, and

necklaces

BG\_TORSO = 5 Worn on the torso, such as armor

BG\_LEFT\_RING = 6 Item worn on left finger (ring)

BG\_RIGHT\_RING = 7 Item worn on right finger (ring)

BG\_QUIVER = 8 Arrows held in quiver

BG\_HEAD = 9 Items worn on the head such as headbands, helms,

etc

BG\_LEGS = 10 Worn on legs, such as greaves

BG\_FEET = 11 Worn on feet, such as boots

BG\_USECODE = 12 Exult-only. Usecode container, used for containing

eggs 'carried' by player

BG\_CLOAK = 13 Exult-only. Items worn around the neck and back

such as cloaks and capes

BG\_GLOVES = 14 Exult-only. Items worn on both hands, such as

gloves

BG\_TRIPLE\_BOLTS = 15 Special slot used by triple crossbow bolts

BG\_EARS = 16 Exult-only. Earrings, such as the serpent earrings

BG\_BACK\_SHIELD = 17 Exult-only. Shield slung across the back

BG\_BACK\_2H = 19 Exult-only. Weapon slung across the back

BG\_BOTH\_HANDS = 20 Special slot for item wielded with both hands

BG\_LRGLOVES = 21 Special slot for item wielded with both ring slots

BG\_AMULET = 22 Exult-only. Items worn on the neck such as a

collar

\*\*\*\* END BG SPECIFIC \*\*\*\*

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

In SI, the 'spot' parameter is limited to the 0-17 range. It can take any of

the following values:

SI\_OFF\_HAND = 0 Items wielded in the off-hand

SI\_SHIELD\_HAND = 0 Off-hand is also called shield hand

SI\_WEAPON\_HAND = 1 Items wielded in the weapon hand

SI\_CLOAK = 2 Items worn around the neck and back such as cloaks

and capes

SI\_AMULET = 3 Items worn on the neck such as a collar

SI\_HEAD = 4 Items worn on the head such as a helm

SI\_GLOVES = 5 Items worn on both hands, such as gloves

SI\_USECODE = 6 Usecode container, used for containing eggs

'carried' by player

SI\_RIGHT\_RING = 7 Ring or item worn on right hand

SI\_LEFT\_RING = 8 Ring or item worn on left hand

SI\_EARS = 9 Earrings, such as the serpent earrings

SI\_QUIVER = 10 Arrows held in quiver

SI\_BELT = 11 Worn across the waist, such as belts and girdles

SI\_TORSO = 12 Worn on the torso, such as armor

SI\_FEET = 13 Worn on feet, such as boots

SI\_LEGS = 14 Worn on legs, such as greaves

SI\_BACKPACK = 15 Containers worn on the back, such as backpacks

SI\_BACK\_SHIELD = 16 Shield slung across the back

SI\_BACK\_2H = 17 Weapon slung across the back

SI\_TRIPLE\_BOLTS = 18 Exult-only. Special slot used by triple crossbow

bolts

SI\_BOTH\_HANDS = 20 Special slot for item wielded with both hands

SI\_LRGLOVES = 21 Exult-only. Special slot for item wielded with

both ring slots

\*\*\*\* END SI SPECIFIC \*\*\*\*

Return:

Returns true if the NPC is wearing an object of the specified shape and

frame on the inspected slot, false if not.

int UI\_get\_weapon(actor npc) [SI, Exult]

int npc->get\_weapon()

Inspect what type of weapon an NPC is using.

Parameters:

npc The NPC whose weapon is to be inspected.

Return:

Zero if no weapon, one- or two-handed, is wielded by the NPC. Otherwise, the

shape in 'SHAPES.VGA' of the weapon being used.

UI\_set\_opponent(actor npc, object obj)

npc->set\_opponent(object obj)

Makes the supplied NPC target the desired object. This does not make the NPC

attack unless his schedule is IN\_COMBAT. This intrinsic will reevaluate if

the NPC's oppressor is actually attacking him, and will invalidate the

oppressor if not.

Parameters:

npc The NPC whose opponent we wish to set.

obj What the target should be set to. If the target is an NPC, his/her

oppressor will be set to the attacking NPC.

UI\_sit\_down(actor npc, object obj)

npc->sit\_down(object obj)

Causes the specified NPC to sit on the given chair. 'npc' sit on 'obj'.

'obj' must be a chair/seat shape (currently, hardcoded to 873/292). For

barge seats, also runs the barge usecode when everyone in the party is

seated.

Parameters:

npc The NPC that is going to sit down.

obj The chair that should be sat upon.

If the chair is shape 292 (a barge seat), if there is a barge containing

the chair and if all party members are sitting, then Exult will run the

barge's usecode (function 0x634).

UI\_get\_body\_npc(object obj) [SI, Exult]

obj->get\_body\_npc()

Inspects what NPC (is associated with the given dead body.

Parameters:

obj The object to be inspected.

Return:

If the supplied object is the dead body of an NPC, returns the NPC's number,

as reported by get\_npc\_num intrinsic call; otherwise, returns zero.

actor UI\_resurrect(object obj)

actor obj->resurrect()

Brings a dead body back to life. If 'obj' is the body of an NPC, resurrects

the NPC and returns a reference to the NPC; otherwise, returns a null ('0')

reference.

Parameters:

obj The body that is to be resurrected. Monsters cannot be brought back

to life in this manner, so it must be the body of a true NPC. This

body will be deleted after the usecode ends, and all of its contents

will be transferred to the resurrected NPC.

Return:

Zero if the supplied object was not the body of an NPC, otherwise an object

reference to the resurrected NPC.

UI\_resurrect\_npc(actor npc) [SI]

npc->resurrect\_npc()

Brings the desired NPC back to life. If the NPC has a body, this body will

be completely ignored; moreover, the NPC will be back without any equipment

that was left on his body when he died.

Parameters:

npc Who is to be resurrected.

UI\_set\_polymorph(actor npc, int shape) [SI, Exult]

npc->set\_polymorph(int shape)

Makes an NPC take a different, temporary shape. The polymorphed shape is

persistent across save games. The NPC can be returned to its normal shape by

unsetting the NPC's POLYMORPH flag with a call to clear\_item\_flag intrinsic.

Parameters:

npc The NPC to be polymorphed.

shape The shape in 'SHAPES.VGA' that the NPC will take.

int UI\_get\_temperature(actor npc) [SI, Exult]

int npc->get\_temperature()

Inspects a NPC's temperature. The NPC's temperature changes over time

depending on whether or not his FREEZE flag is set.

Parameters:

npc The NPC whose temperature is sought.

Return:

A numeric value in the 0-63 range specifying how cold the NPC is; 0 being

normal temperature and 63 being freezing cold.

UI\_set\_temperature(actor npc, int temp) [SI, Exult]

npc->set\_temperature(int temp)

Changes a NPC's temperature. The NPC's temperature changes over time

depending on whether or not his FREEZE flag is set.

Parameters:

npc The NPC whose temperature is to be changed.

temp The new value for the temperature. This is value in the 0-63 range

specifying how cold the NPC is; 0 being normal temperature and 63

being freezing cold.

int UI\_get\_temperature\_zone(object obj) [SI, Exult]

int obj->get\_temperature\_zone()

Inspects a NPC's temperature zone. The temperature zone is a grouping of

temperature ranges that determine how the NPC is affected by the cold. Like

the NPC's temperature, it changes over time depending on whether or not his

FREEZE flag is set.

Parameters:

obj The NPC whose temperature zone is sought.

Return:

Zero if anything but an NPC is supplied; otherwise, a number in the 1-5

range, equal to the frame of the stats gump that should be displayed.

int UI\_get\_npc\_warmth(object obj) [SI, Exult]

int obj->get\_npc\_warmth()

Inspects the warmth given by a NPC's equipment.

Parameters:

obj The object to be checked.

Return:

The combined warmth given by all objects worn by the NPC, or -75 if supplied

with anything but a valid NPC.

================================================================================

PARTY INTRINSICS

================================================================================

These intrinsics relate to obtaining or modifying the list of party members.

actor[] UI\_get\_party\_list()

actor[] UI\_get\_party\_list2()

Gets a list of all party members.

Return:

An array containing object references to all of the party's members,

including the avatar.

UI\_add\_to\_party(actor npc)

npc->add\_to\_party()

Makes the specified NPC join the avatar's party, if there is enough room.

The NPC's schedule is changed to FOLLOW\_AVATAR, and his/her alignment is

changed to GOOD.

Parameters:

npc Who should be added to the party.

UI\_remove\_from\_party(actor npc)

npc->remove\_from\_party()

Removes the specified NPC from the avatar's party. The NPC's alignment is

changed to NEUTRAL.

Parameters:

npc Who should be removed from the party.

object[] UI\_get\_dead\_party(object obj)

object[] obj->get\_dead\_party()

Gets a list of all dead party members near a given object.

Parameters:

obj The object around which dead bodies will be searched.

Return:

An array containing object references to the bodies of all dead party

members within 50 tiles of the given object, or zero if no party members

were found.

================================================================================

AVATAR INTRINSICS

================================================================================

All intrinsics in this section are about obtaining or modifying properties

specific to the Avatar.

actor UI\_get\_avatar\_ref()

Gets an object reference to the avatar.

Return:

The Avatar's object reference.

bool UI\_in\_combat()

Checks if the avatar is in combat.

Return:

true if the avatar is in combat mode, false otherwise.

bool UI\_is\_pc\_female()

Determines the avatar's gender.

Return:

true if the avatar is a woman, false otherwise.

int UI\_get\_skin\_colour() [SI, Exult]

Obtains the avatar's skin color.

Return:

A numeric value describing the avatar's skin color. The return values are

determined by a table in 'avatar\_data.txt'.

bool UI\_is\_pc\_inside()

Determines if the avatar has a roof over his head.

Return:

Returns true if there is a roof above the avatar's head, false otherwise.

bool UI\_wearing\_fellowship()

Determines if the avatar is wearing a fellowship amulet.

Return:

Returns 'true' if the avatar is wearing a fellowship amulet (shape 955,

frame 1), 'false' otherwise.

bool UI\_approach\_avatar(actor npc) [SI, Exult]

bool npc->approach\_avatar()

bool UI\_approach\_avatar(actor npc, int unk1, int unk2)

bool npc->approach\_avatar(int unk1, int unk2)

Makes a NPC approach the avatar at fairly high speed.

\*Warning:\* If the NPC is initially at more than 10 tiles from the avatar,

Exult will let him/her walk for about 1/5 seconds before returning,

effectively halting the game in the mean time.

\*NOTICE:\* In the original, this intrinsic \*seems:\* to create the NPC

off-screen, while it approaches the avatar due to si\_path\_run\_usecode or the

'TALK' schedule. This requires further investigation; this means that \*this

intrinsic is subject to change\*.

Parameters:

npc Who should approach the avatar.

unk1 Unknown parameter from the original. Exult does not use this

parameter, but \*this is subject to change\*. Ignore this parameter at

your own peril.

unk2 Unknown parameter from the original. Exult does not use this

parameter, but \*this is subject to change\*. Ignore this parameter at

your own peril.

Return:

false if the NPC is dead or cannot reach the avatar, true otherwise.

UI\_call\_guards()

Causes guard to come and attack a thieving avatar.

UI\_attack\_avatar()

Causes everyone nearby to attack thieving avatar.

UI\_stop\_arresting()

Any guards approaching the avatar to arrest will stop what they are doing

and will start to wander instead.

UI\_save\_pos(object obj) [SI, Exult]

obj->save\_pos()

Saves the current position/map of an object. This persists across save

games, but there is only one 'slot' in which the position is saved.

Therefore, only the last stored position is available.

Parameters:

obj The object whose position we want.

UI\_teleport\_to\_saved\_pos(object obj, bool flag) [SI, Exult]

obj->teleport\_to\_saved\_pos(bool flag)

Teleports the avatar and party to the last saved position. Note that only

the last position stored by the UI\_save\_pos intrinsic is available.

Parameters:

obj If this is not the avatar, nothing happens. Otherwise, the party is

teleported to the stored position.

flag This optional parameter controls controls whether or party members

that are paralyzed or asleep; if true, these members will be

teleported, otherwise, they won't defaults to false.

================================================================================

WEATHER/TIME INTRINSICS

================================================================================

These intrinsics allow you to check or alter the weather, the game clock and

usecode timers.

int UI\_get\_timer(int timer)

Gets how many hours elapsed since the timer was set.

Parameters:

timer Numeric value indicating the timer to be inspected.

Return:

If the specified timer has been initialized by a previous call to

'UI\_set\_timer' intrinsic, the return is the number of hours since the

initialization (or reinitialization) of the timer. Uninitialized timers

return a random number in the 0-12 hour range.

UI\_set\_timer(int timer)

Resets a timer to the current game time.

Parameters:

timer Numeric value indicating the timer to be initialized.

int UI\_game\_day()

Gets the game day.

Return:

The current game day, starting with day 0.

int UI\_game\_hour()

Gets the game hour.

Return:

The current game hour in 24-hour format.

int UI\_game\_minute()

Gets the game minutes.

Return:

The current game minute (0-59).

int UI\_part\_of\_day()

Gets the number of 3-hour periods that have elapsed in full since midnight.

Return:

Returns an integer in the 0-7 range corresponding to the number of 3-hour

periods that have elapsed in full since midnight. More specifically, the

return value is equal to 'UI\_game\_hour()%3'. The following constants are

defined for convenience:

MIDNIGHT = 0 EARLY = 1 DAWN = 2 MORNING = 3

NOON = 4 AFTERNOON = 5 EVENING = 6 NIGHT = 7

UI\_nap\_time(object obj)

obj->nap\_time()

Makes the avatar lay down to sleep in the specified bed. What constitutes a

bed is currently hard-coded (as of 2009/02/27). Once the avatar reaches the

bed, Exult will call usecode function 0x622. If the bed is occupied, causes

one of the party members to pop up and say so.

Parameters:

obj The bed where the avatar should sleep. This bed should be unoccupied.

UI\_advance\_time(int amount)

Advances the game time forward.

Parameters:

amount Number of game ticks that should be advanced; 25 ticks are equal

to one game minute, and 1500 ticks are equal to one game hour.

UI\_stop\_time(int length)

Stops time for all but the avatar and party.

Parameters:

length How long the effect should last. The actual duration, in seconds,

is equal to one fourth of this value.

int UI\_get\_weather()

Returns the current weather state. Return values are:

Return:

A numeric value indicating what the current weather is. Possible values are:

CLEAR\_WEATHER = 0 No clouds, no precipitation.

SNOWING = 1 Snowing.

RAIN = 2 Raining

SPARKLE = 3 Magical storm, causes magic to fail.

FOG = 4 Nothing happens.

OVERCAST = 5 Light cloud cover.

CLOUDY = 6 Heavy cloud cover.

UI\_set\_weather(int type)

Modifies the current weather to the desired type. The new weather lasts for

15 game minutes.

Parameters:

type Numeric value indicating what the weather should be. Can be any of

the values returned by UI\_get\_weather intrinsic.

================================================================================

USER INTERFACE INTRINSICS

================================================================================

With these intrinsics, you can ask the user for input or display feedback

for his actions.

mixed[4] UI\_click\_on\_item()

Displays the targeting square and waits for the player to click on

something.

Return:

An array with 4 elements. The first element is the object clicked, or zero

if the player clicked a tile; the remaining 3 elements make up the (x, y, z)

location of the object or tile clicked.

If used after a 'set\_intercept\_item' intrinsic call, the preassigned object

or tile is returned in the same manner as if the player had clicked it.

This intrinsic has a special case: when event == WEAPON (weapon hit), and if

item is a valid object, the return will be overridden to be it (although a

'set\_intercept\_item' intrinsic call takes precedence over this special

case).

UI\_set\_intercept\_item(object obj) [Exult]

obj->set\_intercept\_item()

UI\_set\_intercept\_item(int obj[2])

UI\_set\_intercept\_item(int obj[3])

UI\_set\_intercept\_item(mixed obj[4])

Preassigns the return of the next 'UI\_click\_on\_item' intrinsic' call.

Parameters:

obj What we want to set the return of the next the next

'UI\_click\_on\_item' intrinsic' call to. Can be an object reference, a

location (2 or 3 coordinates) or even the return of another

'UI\_click\_on\_item' intrinsic' call.

int UI\_input\_numeric\_value(int min, int max, int step, int default)

Prompts the player for numerical input with the bar slider.

Parameters:

min The lowest number allowed.

max The highest number allowed.

step The distance between any two consecutive numbers that can be

chosen.

default The initial value selected; this value is forced to conform to

the boundaries specified above.

Return:

The number chosen by the player.

bool UI\_mouse\_exists()

Determines if a mouse exists.

Return:

Exult always returns true; the original games will return true if a mouse if

present, false otherwise.

UI\_flash\_mouse(int cursor)

Momentarily changes the mouse cursor to another shape.

Parameters:

cursor Numeric parameter indicating what the new shape will be. Valid

values are:

CURSOR\_X = 1 CURSOR\_OUT\_OF\_RANGE = 2

CURSOR\_OUT\_OF\_AMMO = 3 CURSOR\_TOO\_HEAVY = 4

CURSOR\_WONT\_FIT = 5

UI\_ambient\_light(bool onoff) [SI, Exult]

Toggles on or off a permanent version of the light spell.

Parameters:

onoff Flag specifying whether we want to turn the light on or off: true

to turn it off, false to turn it on.

UI\_cause\_light(int length)

Creates a light which illuminates the screen.

Parameters:

length How long the created light will last. The duration is in game

minutes.

UI\_infravision(actor npc, bool onoff) [SI, Exult]

npc->infravision(bool onoff)

Toggles infravision for the given NPC.

Parameters:

npc The NPC that should have infravision toggled. This intrinsic has

effect only for party members.

onoff Flag specifying whether we want infravision or not: true to turn it

on, false to turn it off.

UI\_set\_light(object obj, bool onoff)

obj->set\_light(bool onoff)

Marks the given light source as being added to or removed from its

containing NPC.

Parameters:

obj The light source that is being added or removed.

onoff Flag specifying whether we want to add or remove the light source:

true to add it to its containing NPC, false to remove it.

UI\_earthquake(int length)

Causes the screen to shake. Also plays the earthquake SFX at the start of

the effect.

Parameters:

length The earthquake's duration, in ticks.

UI\_armageddon()

Ends the world and kills everyone in it. This is the Armageddon spell from

BG; use with care :-). This intrinsic also exists in the original SI, though

unlike what happens in BG, it seems to leave no survivors.

Who survives armageddon seems to be hard-coded in the original games; in

Exult, you can control who survives this intrinsic by setting the 'Survives

Armageddon' flag in the 'NPC Flags' tab in Exult Studio.

Those 'killed' by this intrinsic cannot be resurrected, and they do not

leave lootable corpses.

Monster eggs are disabled, as are intrinsics that call guards to attack a

thieving avatar.

UI\_restart\_game()

Stops the music and restarts the game from the last save. The original games

started from the beginning of the game; Exult does it differently to

preserve the last quicksave.

UI\_run\_endgame(bool flag)

Runs the endgame.

Parameters:

flag Boolean value specifying the manner in which the game was finished.

If it is true, will set the game-completion flag that allows the

endgame to be viewed from the title screen.

\*\*\*\*\*\* BG SPECIFIC \*\*\*\*\*\*

If 'flag' is 'false', the avatar stepped through the Black Gate; if 'true',

the avatar destroyed the Black Gate.

\*\*\*\* END BG SPECIFIC \*\*\*\*

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

'flag' has no effect in the endgame.

\*\*\*\* END SI SPECIFIC \*\*\*\*

UI\_error\_message([special])

Prints the input variables to 'stdout'.

Parameters:

One to twelve parameters of any type, whether or not they are arrays. Please

note that the original games only one parameter was accepted.

string UI\_printf(string out[]) [Exult]

Prints formatted text to 'stdout'.

Parameters:

out An array of strings. The first element is the 'format' string. The

format string gives the basic text, and has a placeholder for each of

the other elements of the array; each element past the first writes

its text at the next available placeholder. Placeholders are the '%s'

sequence, without the single quotes, embedded in the format string.

Return:

An empty string.

================================================================================

DISPLAY INTRINSICS

================================================================================

All intrinsics in this section are related to showing things to the player,

be they other portions of the game map or objects from 'SPRITES.VGA'.

UI\_center\_view(object obj) [Exult]

obj->center\_view()

Centers the screen on a given object.

Parameters:

obj The object that the screen should be centered on.

UI\_display\_area(int pos[])

Displays the specified location on-screen. Overlays sprite shape 10 over the

screen.

Parameters:

pos The location about which the screen is to be centered. Can be a

3-element (x, y, z) location or a 4-element (x, y, z, map) location.

If map is omitted or is -1, the current map is used.

UI\_set\_camera(object obj)

obj->set\_camera()

Centers the screen on a given object.

Parameters:

obj The object that the screen should be centered on. If it is an NPC,

the camera will follow that NPC.

UI\_view\_tile(int[2] pos)

Centers screen around a given tile. It is usually better to use

UI\_display\_area instead.

Parameters:

pos The (x, y) location to be displayed.

UI\_wizard\_eye(int length)

Lets the player move the screen around, keeping the avatar in its present

location. Overlays sprite shape 10 over the screen.

Parameters:

length The duration of the effect, in ticks. Actual duration is increased

by 50% in Exult.

UI\_fade\_palette(int length, int unk, bool type)

Causes the screen to fade to/from black and plays music.

Parameters:

length The length, in ticks, of the fade; that is, how long it will take

for the fade to happen.

unk Reserved; unknown. This is an unknown parameter from the original

games; it is unimplemented in Exult. It cannot be left out,

however.

type The type of fade. If true, fades from black; if false, fades to

black.

UI\_fade\_palette\_sleep(int length, int unk, bool type) [SI, Exult]

Causes the screen to fade to/from black.

When fading to black, plays music 24; otherwise, plays music 22.

Parameters:

length The length, in ticks, of the fade; that is, how long it will take

for the fade to happen.

unk Reserved; unknown. This is an unknown parameter from the original

games; it is unimplemented in Exult. It cannot be left out,

however.

type The type of fade. If true, fades from black; if false, fades to

black.

UI\_set\_time\_palette()

Resets palette to default palette, based on time of day and number of

lights.

UI\_item\_say(object obj, string bark)

obj->item\_say(string bark)

Causes text to be displayed near the object.

Parameters:

obj The object which is to display the text.

bark Text to be displayed.

UI\_clear\_item\_say(object obj) [SI, Exult]

obj->clear\_item\_say()

Deletes the string (if any) being displayed near the desired object.

Parameters:

obj The object in question.

UI\_display\_runes(int gump, string runes)

Displays the selected gump on screen and prints lines of runic text

on-screen. What font is used to draw the text depends on the gump shown, and

is currently hard-coded (as of 2009/02/26).

Parameters:

gump The shape in 'GUMPS.VGA' that will be used.

runes An array containing one element per line of runes to display.

If the avatar's READ flag is set, some symbols are converted into pairs of

letters. Specifically:

( TH

) EE

\* NG

+ EA

, ST

Additionally, all lowercase letters will be turned into uppercase letters,

and pipes ('|') will be converted to spaces.

When compiling code, UCC accepts some escape sequences to make these strings

easier to read:

( \{th}

) \{ee}

\* \{ng}

+ \{ea}

, \{st}

| \{dot}

UI\_book\_mode(object obj)

obj->book\_mode()

Displays a book or scroll on screen. Text in the book or scroll is supplied

through normal conversation methods.

Parameters:

obj The object whose shape is to be used to determine what shape from

'GUMPS.VGA' is to be shown. This association is currently hard-coded

(as of 2009/02/27): if the shape of the supplied object is equal to

797, a scroll gump will be displayed; otherwise, a book gump will be

displayed.

\*\*\*\*\*\* SI SPECIFIC \*\*\*\*\*\*

There is one additional shape which displays a scroll gump: shape 707. This

shape, and shape 705 (serpentine book), will both display text using

serpentine characters; if the avatar's 'READ' flag is set, any serpentine

runes will be translated into normal runes.

\*\*\*\* END SI SPECIFIC \*\*\*\*

UI\_book\_mode\_ex(bool is\_scroll, int font) [Exult]

UI\_book\_mode\_ex(bool is\_scroll, int font, int gump)

Displays a book or scroll on screen. Text in the book or scroll is supplied

through normal conversation methods.

Parameters:

is\_scroll Flag determining whether to display a scroll (true) or a book

gump (false). This determines the area occupied by the text,

and will determine what shape from 'GUMPS.VGA' to display if

none is specified.

font Integer that determines what shape from 'FONTS.VGA' will be

used to render the text.

gump Numerical value determining what shape from 'GUMPS.VGA' will be

used. The text area is not affected by this parameter.

\*\*\*\* IMPORTANT NOTES \*\*\*\*

Unlike the 'UI\_book\_mode' intrinsic, this intrinsic performs \*no\*

translation of any kind if the avatar's 'READ' flag is set. This is an

intentional design feature -- for example, text written in Gargish and being

displayed in a Gargish font should be translated by more than a mere font

substitution. Also, the gump shapes should be 'compatible' with the standard

book/scroll shapes as Exult will write the text in the same region of the

screen as it does for standard books/scrolls.

\*\* END IMPORTANT NOTES \*\*

UI\_close\_gumps()

Closes all gumps.

\*\*\*\* IMPORTANT NOTES \*\*\*\*

This intrinsic does nothing if called when event == READIED or event ==

UNREADIED. See 'UI\_close\_gumps2' if you need it to work in these cases.

\*\* END IMPORTANT NOTES \*\*

UI\_close\_gumps2() [Exult]

Closes all gumps.

UI\_close\_gump(object obj)

obj->close\_gump()

Closes an object's gump.

Parameters:

obj The object whose gump is to be closed.

\*\*\*\* IMPORTANT NOTES \*\*\*\*

This intrinsic does nothing if called when event == READIED or event ==

UNREADIED. See 'UI\_close\_gump2' if you need it to work in these cases.

\*\* END IMPORTANT NOTES \*\*

UI\_close\_gump2(object obj) [Exult]

obj->close\_gump2()

Closes an object's gump.

Parameters:

obj The object whose gump is to be closed.

bool UI\_in\_gump\_mode()

Returns 'true' if showing gumps, 'false' otherwise.

UI\_display\_map() [BG]

Displays the main game map. Will also display the sextant crosshairs if the

party has a sextant.

UI\_si\_display\_map(int num) [SI]

Displays the specified map on screen.

Parameters:

num Index of the map to be displayed. This is converted into a shape in

'SPRITES.VGA' in a hard-coded manner.

UI\_display\_map\_ex(int mapshp, bool show\_loc) [Exult]

Displays the given map centered in the screen.

Parameters:

mapshp The shape in 'SPRITES.VGA' to be displayed as a map.

show\_loc true to show the sextant crosshairs, false otherwise. Showing

the crosshairs does not depend on being underground or actually

having sextants.

UI\_lightning()

Causes a lightning effect. The screen flashes, and the corresponding sound

effect is played.

UI\_sprite\_effect(int sprite, int x, int y, int vel\_x, int vel\_y, int frame, int

reps)

Displays a graphical effect at the given location.

Parameters:

sprite The shape in 'SPRITES.VGA' to be displayed.

x The initial horizontal location at which the sprite will be

displayed.

y The initial vertical location at which the sprite will be

displayed.

vel\_x The horizontal speed at which the sprite will move. This is added

to the location at every tick.

vel\_y The vertical speed at which the sprite will move. This is added to

the location at every tick.

frame The initial frame in 'SPRITES.VGA' of the sprite. This is

increased by one every tick, and may loop back to zero when it

reaches the last frame (depending on the value of the 'reps'

parameter).

reps Number of repetitions to execute. If zero, the sprite will be

displayed for a tick and then go away. If positive, how many times

the sprite frame will go through all of its frames. If negative,

the sprite will go through all of its frames \*once\*.

UI\_obj\_sprite\_effect(object obj, int sprite, int rel\_x, int rel\_y, int vel\_x,

int vel\_y, int frnum, int reps)

obj->obj\_sprite\_effect(int sprite, int rel\_x, int rel\_y, int vel\_x, int vel\_y,

int frnum, int reps)

Creates a sprite effect from 'SPRITES.VGA'. This sprite effect is tied to

'obj'; if 'obj' moves, the sprite will move along. 'sprite' is the sprite's

shape number, ('rel\_x', 'rel\_y') is the relative (x, y) offset of the sprite

relative to the position of 'obj', ('vel\_x', 'vel\_y') is the (x, y) speed

(i.e., change in coordinates at every tick, and in addition to any motion of

'obj'). 'frnum' is the initial frame of the sprite effect. If 'reps' < 0,

the sprite effect goes through all frames (at the rate of one frame per

tick) and deletes itself; otherwise, 'reps' is the total number of ticks the

animation will last before deleting itself, looping the sprite's frames if

it is needed.

Displays a graphical effect around a given object. This sprite effect is

linked to the object, and will move with it.

Parameters:

obj The object about which the effect will happen.

sprite The shape in 'SPRITES.VGA' to be displayed.

rel\_x The initial horizontal offset from the object at which the sprite

will be displayed.

rel\_y The initial vertical offset from the object at which the sprite

will be displayed.

vel\_x The horizontal speed at which the sprite will move. This is added

to the horizontal offset at every tick.

vel\_y The vertical speed at which the sprite will move. This is added to

the vertical offset at every tick.

frame The initial frame in 'SPRITES.VGA' of the sprite. This is

increased by one every tick, and may loop back to zero when it

reaches the last frame (depending on the value of the 'reps'

parameter).

reps Number of repetitions to execute. If zero, the sprite will be

displayed for a tick and then go away. If positive, how many times

the sprite frame will go through all of its frames. If negative,

the sprite will go through all of its frames \*once\*.

================================================================================

OTHER INTRINSICS

================================================================================

All intrinsics that did not fit in the previous categories.

bool UI\_add\_spell(int index, int unk, object obj)

Adds a spell to a spellbook.

Parameters:

index The spell to be added. This index starts at zero, and must be in

the 0-71 range.

unk Reserved, must be zero. This is an unknown parameter from the

original games; it doesn't seem to have any effects there, and is

unused in Exult. You must still pass it to calls to this intrinsic,

though.

obj What spellbook the spell should be added to.

Return:

false if the supplied object is not a spellbook or if it already has the

spell; true otherwise.

bool UI\_has\_spell(object obj, int index) [Exult]

Checks if a spellbook has one particular spell.

Parameters:

obj The spellbook to be inspected.

index The spell to be checked. This index starts at zero, and must be in

the 0-71 range.

Return:

true if the supplied object is a spellbook and contains the spell; false

otherwise.

bool UI\_remove\_spell(object obj, int index) [Exult]

Removes the specified spell from the desired spellbook.

Parameters:

obj The spellbook that will be modified.

index The spell to be removed. This index starts at zero, and must be in

the 0-71 range.

Return:

true if the supplied object is a spellbook and had the spell; false

otherwise.

UI\_remove\_all\_spells(object obj) [SI, Exult]

obj->remove\_all\_spells()

Removes all spells from the desired spellbook.

Parameters:

obj The spellbook to be cleared. This intrinsic has no effect if this is

not a valid spellbook.

int UI\_get\_array\_size([special])

Gets how many elements an array has.

Parameters:

[special] A single array of any type or size.

Return:

If supplied with a valid array, returns the number of elements in the array;

otherwise, returns 1.

UI\_mark\_virtue\_stone(object obj) [BG]

obj->mark\_virtue\_stone()

Imprints a virtue stone with its current location. Any subsequent calls to

recall\_virtue\_stone that target that same stone will teleport the avatar's

party to this stored location. The stored location includes the map where

the stone is.

Parameters:

obj The virtue stone to be marked.

UI\_recall\_virtue\_stone(object obj, bool flag) [BG]

obj->recall\_virtue\_stone(bool flag)

Teleports the party to the virtue stone's stored location. The stone is

forcibly given to the avatar if needed.

Parameters:

obj The virtue stone that contains the destination.

flag This optional parameter controls controls whether or party members

that are paralyzed or asleep; if true, these members will be

teleported, otherwise, they won't defaults to false.

UI\_set\_orrery(int pos[3], int state) [BG]

Sets the position of the planets in the orrery at Moonglow.

Parameters:

pos The location of the orrery.

state Numeric value in the 0 to 9 range determining the relative

positions of the planets.

The intrinsic tries to find 'Planet Britannia' near the specified location;

it will delete the nearby planets, if any, and create new ones in the

required positions.

int UI\_is\_water(object param)

int param->is\_water()

int UI\_is\_water(int param[2])

int UI\_is\_water(int param[3])

int UI\_is\_water(mixed param[4])

Determines if a given tile or object is a water tile.

Parameters:

param The object or location to be tested.

The parameter can be an object, an array with 2 components, an array with 3

components, or an array with 4 components. The interpretation of the

parameter is as follows:

Type Interpretation/Format

object Exult specific: The parameter is an object or NPC, whose

shape is tested for the water flag.

int [2] The parameter is a (x, y) position specifying a tile to be

tested.

int [3] The parameter is a (x, y, z) position, such as the return

of a UI\_get\_object\_position intrinsic call. Of this

position, (x, y) specifies a tile to be tested.

mixed [4] Exult specific: The parameter is of the form (object, x, y,

z), such as the return of a call to UI\_click\_on\_item

intrinsic. If object is not null, it's shape is tested for

the water flag. Otherwise, (x, y) is used to find and test

the corresponding tile.

Return:

true if the specified tile or object is a water tile, false otherwise.

string UI\_a\_or\_an(string text)

Checks whether a bit of text starts with a vowel or not.

Parameters:

text The string being inspected.

Return:

If the first letter of the supplied string is in the "aeiouyAEIOUY" set,

returns 'an'; otherwise, returns 'a'.

bool UI\_is\_dest\_reachable(actor npc, int pos[]) [Exult]

bool npc->is\_dest\_reachable(int pos[])

Checks if a NPC can walk from its present location to a specified position.

Parameters:

npc The NPC we wish to check.

pos Where we want check if the NPC can go.

Return:

true if the NPC can walk to the supplied destination, false otherwise.

bool UI\_can\_avatar\_reach\_pos(int pos[]) [SI]

Checks if the avatar can walk from its present location to a specified

position. This is similar to, but more limited than, the is\_dest\_reachable

Exult intrinsic.

Parameters:

pos Where we want check if the avatar can go. Must have at least two

elements; if it has exactly two elements, Exult will assume the z

coordinate of the destination to be zero.

Return:

false if the supplied position is invalid or if the avatar cannot reach the

supplied destination, true otherwise.