

Ultima VII Routine and Function declarations.

To be implemented in AGIL

As of 8-2-91

Note: All get and set functions must be able to access and modify items on the "Ethereal Void" stack
Functions marked with an * are for usecode use specifically Some disposable variables do not have the first letter capitolized in the examples, this is meerely a matter of style and should not be used in place of the conversation convention of capitolizing all variables.

Functions:

~GetParty();

Parameters: None

Function: Creates a list of the members of the current party in the form of a list with the Avatar as the first element of the list

Returns: A list of the numbers of the members of the party.

Example: party = ~GetParty(); // party now contains a list of the numbers of the members of the party

~GetParName();

Parameters: None

Function: Creates a list of the members of the current party in the form of a list with the Avatar as the first element of the list

Returns: A list of the strings of the members of the party.

Example: Party = ~GetParName(); // party now contains a list of the names of the members of the party

~GetNPCName(NPC#);

Parameters: NPC# is a valid number for an NPC

Function: Converts the NPC number into a string containing the name of the NPC

Returns: A string containing the name of the NPC specified.

Example: Dupre = ~GetNPCName(_Dupre); // Variable Dupre now contains string "Dupre"

~AvatarName();

Parameters:

Function: Gets the string for the Avatar's name

Returns: A string containing hte Avatar's name

Example: Avatar = ~Avatarname();

~GetFrame(Item#);

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Function: Gets the current frame of the particular instance of an item, NPC, or terrain piece.

Returns: An integer containing the current frame

Example: BobFrame = ~GetFrame(_Bob); // BobFrame now contains a number from 0 to 31 which is the current frame of Bob (note: if Bob is animating, this must be checked by using IsAnim());

~SetFrame(Item#,Frame#);

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Frame# is a valid number between 0 and 31

Function: Sets the frame of a particular instance of an item, NPC or terrain piece to the specified frame IF that is a valid frame for that particular item.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~SetFrame(_Bob, 1);

~CountItem(NPC,Item,Quality,Frame);

Parameters: NPC# is a valid number for an NPC

Item# is a valid number of the instance of an item, NPC, or terrain piece.

Quality is a valid number to be set as the quality for the item

Frame# is a valid number between 0 and 31

Function: Counts occurrences of an item OR returns quantity of a stackable item

CAN ACCEPT _ALL(-3) to search all frames, qualities, quantities of a type

CAN ACCEPT _PARTY(-1) To search party for item

Returns: The number of items found.

Example: allthepartygold=~CountItem(_Party,_Gold,_All,_All);

~GetQuantity(Item#); *

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Function: Gets the current quantity of the particular instance of an item, NPC, or terrain piece.

Returns: An integer containing the current Quantity.

Example: BobQuantity=~GetQuantity(_Bob); // BobQuantity now contains a number from 0 to 255 which is the current quantity of Bob.

~SetQuantity(Item#,Quantity);*

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Quantity is a valid number to be set as the quantity for the item

Function: Sets the Quantity of a particular instance of an item, NPC or terrain piece to the specified Quantity.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~SetQuantity(_Gold, 1);

~GetQuality(Item#);

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Function: Gets the current quality of the particular instance of an item, NPC, or terrain piece

Returns: An integer containing the current Quality.

Example: BobQuality=~GetQuality(_Bob); // BobQuality now contains a number from 0 to 255 which is the current quality of Bob.

~SetQuality(Item#,Quality);

Parameters: Item# is a valid number of the instance of an item, NPC, or terrain piece.

Quality is a valid number to be set as the quality for the item

Function: Sets the Quality of a particular instance of an item, NPC or terrain piece to the specified Quality.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~SetQuality(_Gold, 1);

~PushItem(Source,Item#);

Parameters: Source is a list containing either the X,Y,Z location of an object, the NPC's to scan for the item, _FIND(-4) to force it to find the instance of the item specified(failing if item is not in region), _Party(-1), or _Create(-2).

Item# is a valid number of the type or instance of an item or NPC.

Function: Pushes the specified item (or instance of an item) to the top of the "Ethereal Void" stack. This can be done either to create an item or to remove an item from the "world" for manipulation or destruction.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~PushItem(_Avatar,#Green_Potion)

~PopToNPC(Item#);*

Parameters: Item# is a valid number of the instance of an item(container), NPC ,
_Destroy/_Create(-2) or _Party(-1).

Function: Moves the top item off of the EV stack to the specified NPC or container or destroys it.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~PopToNPC(_Avatar);

~PopToMap(Destination);

Parameters: Destination is a list containing either the X,Y,Z location of an object or _Destroy(
1) An item's Z coord is it's level; it will be sorted on top of any other items at
that X,Y on level (i.e. 1st, 2nd, 3rd)

Function: Places the top item on the EV stack at the specified location or destroys it.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~PopToMap(coordlist);

~GetStat(NPC#,Statistic);

Parameters: NPC# is a valid number for an NPC

Statistic is a member for the following list of valid statistics for NPCs:

_Str, _Dex, _Int, _Hits, _MaxHits, _Cbt, _Mag,
_MaxMagic, _TrainingPoints

Function: Gets the desired statistic of the requested NPC.

Returns: An integer containing the value requested.

Example: Bobdex=~GetStat(_Bob, _Dex);

~SetStat(NPC#,Statistic,Value);

Parameters: NPC# is a valid number for an NPC

Statistic is a member of the following list of valid statistics for NPC's

_Str, _Dex, _Int, _Hits, _MaxHits, _Combat, _Magic,
_MaxMagic, _TrainingPoints

Value is a number + or - 1 to 255

Function: Alters the value of the statistic desired by the requested Increment.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~SetStat(_Bob,1)

~IsTypeNear(Item#);

Parameters: Item# is a valid number of a type

Function: Determines if the desired item is on screen

Returns: A logical whose value is determined by the function (TRUE or FALSE)

Example: check=~IsTypeNear(_Grass);

~IsNPCNear(NPC#);

Parameters: NPC# is a valid number of an NPC

Function: Determines if the desired NPC is on screen

Returns: A logical whose value is determined by the function (TRUE or FALSE)

Example: check=~IsNPCNear(_Iolo);

~GetNPCStatus(NPC#);

Parameters: NPC# is a valid number for an NPC or Creature.

Function: Determines the status bits of the desired creature

Returns: a list of the possible statuses which apply to the NPC.(see status bits for possible

statuses)

Example: Bobstat=~GetNPCStatus(_Bob);

~IsInInv(Subject,Quantity,Item,Quality,Frame);

Parameters: Subject is a list of NPCs or creatures

Item is a valid Type number

Quantity is a valid number to be set as the quantity for the item to be checked

Quality is a valid number to be set as the quality for the item to be checked

Frame is a valid number for a existing frame for that object.

NOTE: Quantity, quality and frame can take an _ALL(-3) parameter, meaning that item's identifier is to be ignored in this search; i.e., gold will not need to check for frame or quality but must still have a value for the function call.

Function: Scans the inventory of the list of NPCs to find if the item exists

Returns: A logical, true if the item is found; otherwise false.

Example: Bobgotit=~IsInInv(_Bob);

~AskYesNo();

Parameters:

Function: Pull up a mini yes/no gump and get the required answer.

Returns: A logical equivalent to the answer received by the function.

Example: DoYouWantIt=~AskYesNo(); // DoYouWantIt contains either true or false

~AskBarInput(First#,Last#,Step);

Parameters: First# is the lowest number on the input bar.

Last# is the highest number on the input bar.

Step is the increment which the input bar will be able to determine.

Function: Puts up a gump which will take numeric input from the user

Returns: An integer which contains the value returned by the user

Example: HowManyOfTheseDoYouWant=~AskBarInput(1,10,1);

~AskParty();

Parameters:

Function: Brings up a gump listing all the members of the party and allows the player to select one of them.

Returns: A number corresponding to the NPC selected by the user.

Example: whoisit=~AskParty();//if user selects Avatar, whoisit now equals Avatar's number.

~AskList(List);

Parameters: List is a list of strings to be selected from by the users.

Function: Allows the user to select from a list of strings.

Returns: The string selected by the user.

Example: whatdoyouwant=~AskList(<<"sex","gold","power">>+listofotherstuff);

~AskOrdList(List);

Parameters: List is a list of numbers to be selected from by the users

Function: Allows the user to select from a list of numbers.

Returns: An integer showing position on list of chosen number

Example: Ref=~AskOrdList(<<"me","you","Bob">>);

~LordOrLady();

Parameters:

Function: Returns a string containing the proper address for the Avatar's gender in "Britannia" mode, i.e., milord or milady

Returns: A string "milord" or "milady"

Example: D00d=~LordOrLady();// D00d gets "milord" or "milady"

~AvatarSex();

Parameters:

Function: Determines gender of the avatar

Returns: Boolean variable for which true is the female case

Example: D00d=~AvatarSex(); // D00d gets either True or False

~GetTime();

Parameters:

Function: Determines the current time in Cycles(3 hours) in the game.

Returns: A number between 0 and 7 corresponding to the time cycle of the day.

Example: Time=~GetTime();// if time = 3am , Time=1

~GetHour();

Parameters:

Function: Determines the current time in hours in the game.

Returns: A number between 0 and 23 corresponding to the hour of the day.

Example: Time=~GetHour();// if time =, 3am Time=3

~GetClock(Clockname);

Parameters: Clockname is a valid name for a predefined timer

Function: Checks for a valid timer which is a piece of memory which counts the hours since the timer was last set to 0; if no such timer exists it returns -1.

Returns: The number of hours since the timer was last set, or -1 if the timer was never initiated

Example: howlongsincethiswasdone=GetClock(_This);// returns in hours how long since the _This timer was set.

~SetClock(Clockname);

Parameters: Clockname is a valid name for a timer.

Function: Clockname is a valid name for a predefined timer. If timer is not defined it generates the timer specified. In either case the specified timer is reset to 0.

Returns: True if timer already existed; false if creates the timer.

Example: Check=~SetClock(this);//this now ==0

~IsContents(Container,Item,Quality,Frame);

Parameters: Container is a valid number for the instance of a container.

Item is a valid Type number.

Quality is a valid number for the item searched for.

Frame is a valid number between 0 and 31 for the frame of the item desired.

Any of the parameters EXCEPT CONTAINER can be passed the _All(-3) parameter so that qualifier will not be essential to the search.

Function: Searches the specified container for the number of the desired item.

Returns: True if container contains the item; False if not .

Example: if ~IsContents(Bag,gold,_all,_all) {[it be full o gold]};

~GetContents(Container,Quantity,Item,Quality,Frame);

Parameters: Container is a valid number for the instance of a container.

Quantity is a valid number (0+) for the number of the item searched for which you are searching.

Item is a valid Type number

Quality is a valid number for the Container or the item searched.
Any of the parameters EXCEPT CONTAINER can be passed the _All(-3) parameter so that qualifier will not be essential to the search.

Function: Searches the specified container for the number of the desired item.

Returns: A list of the occurrences of the item or if item is _all a list of the contents of the container.

Example: Bobsstuff=~GetContents(bag,_all,_all,_all,_all);// returns a list of the instance numbers of the contents of the bag.

~GetCoord(Item); *

Parameters:Item is a valid number of the instance of an item, NPC, or terrain piece

Function: Finds the coordinate for the requested item

Returns: A list of the coordinates in X,Y,Z order

Example: Bobscood=~GetCoord(_Bob);

~GetDist(Item1,Item2); *

Parameters: Item1 and item2 are valid numbers of the instances of an item, NPC, or terrain piece

Function: Determines the distance between Item1 and item 2.

Returns: An integer containing the distance between the objects.

Example: distfroma2b=~GetDist(a,b);

~GetDir(Item1,Item2); *

Parameters: Item1 and Item2 are valid numbers of the instances of an item, NPC, or terrain piece

Function: Determines the direction of item 2 from item 1 in the following format

123
804
765

Returns: An integer referring to the direction of the item per previous chart.

Example: Given:: item 2 is due west of item 1
dir122=~GetDir(Item1,item2);// dir122=8

~IsAnim(Item); *

Parameters: Item is a valid number of the instance of an item, NPC, or terrain piece.

Function:Determines if a given item is animating.

Returns: A boolean value True if item is animating.

Example: isit=~IsAnim(_Millwheel);//isit tells if wheel is animating.

~IsDead(NPC#);

Parameters: NPC# is a valid number for an NPC.

Function: Determines if the specified NPC is dead.

Returns: A boolean variable True if NPC is dead

Example: if (~IsDead(_Iolo)) {[Iolo is dead];}

~SetAnim(Item,State);*

Parameters: Item is a valid number of the instance of an item, NPC, or terrain piece.

State is a boolean value True to start animation False to halt it.

Function: Sets the animating bit to the desired state.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~SetAnim(_Millwheel,_TRUE);//Turns on animation

~GetTarget();*

Parameters:

Function: Returns the instance number of the next item selected.

Returns: Instance number of item selected

Example: mynum=~GetTarget();//the next item selected via the user interface is returned

~Sfx(SFXnumber);

Parameters: SFXnumber is a valid sound effect number.

Function: Runs the specified sound effect. If no sound effects are active it returns false.

Returns: True if operation is performed successfully; False if an error occurs.

Example: check=~Sfx(_Harp+1); // plays second harp sound

~GetMusic();

Parameters:

Function: Finds the current tune being played by the music driver.

Returns: The number of the current music piece.

Example: oldtune=~GetMusic();

~GetAllInRegion(Type); *

Parameters: Type is a valid type for a NPC, monster, terrain piece or item.

Function: Searches region for all occurrences of specified type.

Returns: A list of all the instance numbers of the specified items.

Example: BeesToDie=~GetAllInRegion(_Bee);

~GetWorkType(NPC);

Parameters: NPC is a valid number for an NPC.

Function: Determines the work type of the specified NPC

Returns: The number of the current worktype for the specified NPC.

Example: whatisBobdoing=~GetWorkType(_Bob); //returns what Bob is set to be doing

~
();

Parameters:

Function:

Returns:

Example:

Routines:

SetSpeaker(NPC#);

Parameters: NPC# is a valid number for an NPC.

Function: Brings up a portrait for a character if one is not already on screen, and make the following text appear as if it belonged to the NPC. If the portrait is already loaded, it switches the text output to match the desired NPC.

Returns:

Example: SetSpeaker(_Iolo);

CloseSpeaker(NPC#);

Parameters: NPC# is a valid number for an NPC.

Function: Removes the picture and the text of the specified NPC (do not use this on currently active speaker)(max of three portraits at a time)

Returns:

Example: CloseSpeaker(_Iolo);

TurnOn(< < "keyword", "keyword", ... > >);

Parameters: Keyword is a list of strings to be placed on menu of items for the user to select from

Function: places a word or words on the menu of active key words

Returns:

Example: TurnOn(< < "hi", "bye" > >);

TurnOff(< < "keyword", "keyword", ... > >);

Parameters: Keyword is a list of strings to be removed from menu of items for the user to select from

Function: Removes a word or words from the menu of active key words

Returns:

Example: TurnOn(< < "hi", "bye" > >);

SetWorkType(NPC#,Mode);

Parameters: NPC# is a valid number for an NPC

Function: changes the work type of the desired NPC to the specified work type.

Returns:

Example: [Bob just killed Iolo, how sad]
SetWorkType(_Iolo,_Dead);

JoinParty(NPC#);

Parameters: NPC# is a valid number for an NPC

Function: Sets the NPC into party mode.

Returns:

Example: [Bob, lends his strong arm to your quest]
JoinParty(_Bob);

LeaveParty(NPC);

Parameters: NPC is a valid number for an NPC

Function: Removes the NPC from party mode

Returns:

Example: [Bob thinks you are a fuddy duddy and flips you the finger];
LeaveParty(Bob);

ClearKeys();

Parameters:

Function: Clears all keywords from the user interface

Returns:

Example: TurnOn("hi");// keywords=hi
ClearKeys(); //no key words

PushKeys();

Parameters:

Function: Temporarily sets aside keywords to make room for a new set. these can be retrieved with the popkeys function.

Returns:

Example:

PopKeys();

Parameters:

Function: returns the keys removed by popkeys.

Returns:

Example: TurnOn("baa");//key baa is up
PushKeys();//key baa is gone
TurnOn("moo");//key moo is up
PushKeys(); // both baa and moo are gone
PopKeys();// moo is back
PopKeys();// both moo and baa are back

SetMusic(Music#);

Parameters: Music# is the number of the required music to be played

Function: Loads a new tune into the Music Driver

Returns:

Example: check=SetMusic(_somerealneattune);

PostAction(Item,Script);

Parameters: Item is a legal number for an Item, NPC, creature or terrain piece.

Script is a string of legal commands seperated by + signs and ending with +aEND defining the action for the specified item.

Function: Posts a specified script to the action queue.

Returns:

Example: PostItem(_BigBadMageFromHell,"aUP,aOUT,aEND);