Ultima VII Routine and Function declarations

To be implemented in AGIL.
As of 08-02-1991

Note: All get and set functions must be able to access and modify items on the "Ethereal Void" stack. Functions marked with an * are individually disposable variables that do not have the first letter capitalized in the examples. This case is merely a matter of style and should not replace the conversation convention of capitalizing all variables.

Functions

~GetParty();

Parameter(s): 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();

Parameter(s): 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);

Parameter(s): 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();

Parameter(s):
function: Gets the string for the Avatar's name
Returns: A string containing the Avatar's name
Example:

```
avatar = ~Avatarname();
```

~GetFrame(Item);

Parameter(s): 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();
```

Ultima VII Routine and Function declarations

~SetFrame(Item, Frame);

Parameter(s): 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 specific item.

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

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

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

Parameter(s): 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 the 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 the item

Returns: The number of items found.

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

~GetQuantity(Item); *

Parameter(s): 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); *

Parameter(s): 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 object, NPC or terrain piece to the specified Quantity.

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

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

~SetQuality(Item, Quality);

Parameter(s): 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 object, NPC or terrain piece to the specified Quality.

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

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

Ultima VII Routine and Function declarations

~PushItem(Source, Item);

Parameter(s): 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 thing specified (failing if the item is not in region), Party(-1), or _Create(-2).** Item is a valid number of the type or instance of an object or NPC.

Function: Pushes the specified item (or instance of an object) 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 the operation is performed successfully; False if an error occurs.

Example:
```
check = ~PushItem(_Avatar, Green_Potion);
```

~PopToNPC(Item); *

Parameter(s): 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 the operation is performed successfully; False if an error occurs.

Example:
```
check = ~PopToNPC(_Avatar);
```

~PopToMap(Destination);

Parameter(s): Destination is a list containing either the X, Y, Z location of an object or _Destroy(1) An item's Z coordinate is its level and sorted on top of any other items at that X, Y on the 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 the operation is performed successfully; False if an error occurs.

Example:
```
check = ~PopToMap(coordlist);
```

~GetStat(NPC, Statistic);

Parameter(s): NPC is a valid number for an NPC
Statistic is a member of 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,_MtucHits,_Combat,_Magie, _MaxMagic,_TrainingPoints
Value is a number + or - I to 255

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

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

Example:
```
check = ~SetStat(_Bob,1);
```

~IsTypeNear(Item);

Parameter(s): 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);
```

Ultima VII Routine and Function declarations

~IsNPCNear(NPC);

Parameter(s): 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);

Parameter(s): 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);

Parameter(s): Subject is a list of NPCs or creatures
Quantity is a valid number to be set as the quantity for the item to *be* checked
Item is a valid Type number
Quality is a valid number to be set as the quality for the item to be checked
Frame is a valid number for an existing frame for that object. NOTE: Quantity, quality and frame can take an _ALL(-3) parameter, meaning that item's identifier should 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:
```
bobHasIt = ~IsInInv(_Bob);
```

~AskYesNo();

Parameter(s):
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);

Parameter(s): 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 sump 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();

Parameter(s):
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.
```

Ultima VII Routine and Function declarations

~AskList(List);

Parameter(s): 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 chosen by the user.
Example:
```
whatDoYouWant = ~AskList(<<"sex","gold","power">>
+listofotherstuff);
```

~AskOrdList(List);

Parameter(s): 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();

Parameter(s):
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();

Parameter(s):
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();

Parameter(s):
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();

Parameter(s):
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 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.
```

Ultima VII Routine and Function declarations

~SetClock(Clockname);

Parameter(s): Clockname is a valid name for a timer.

Function: Clockname is a valid name for a predefined timer. If the timer is not defined, it generates the specified timer. The specified timer defaults to 0.

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

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

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

Parameter(s): 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. Note: Any parameters EXCEPT CONTAINER can be passed the _AI1(-3) parameter so that the qualifier is optional to the search.

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

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

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

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

Parameter(s): 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 seeking.
Item is a valid Type number
Quality is the correct number for the Container or the item searched.
Note: Any parameters EXCEPT CONTAINER can be passed the _All(-3) parameter so that qualifier is optional to the search.
Frame is a valid number between 0 and 31 for the frame of the item desired.

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

Returns: A list of the occurrences of the item. If the item is _all returns 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);

Parameter(s): 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:
```
bobsCoord = ~GetCoord(_Bob);
```

~GetDist(item1, Item2); *

Parameter(s): Item1 and Item2 are valid numbers of the instances of an item, NPC, or terrain piece

Function: Determines the distance between Iteml and item 2.

Returns: An integer containing the distance between the objects.

Example:
```
distanceFromA2B = ~GetDist(a, b);
```

Ultima VII Routine and Function declarations

~GetDir(Item1, Item2); *

Parameter(s): 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);

Parameter(s): 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 the item is animating.

Example:

```
isAnimating = ~IsAnim(_Millwheel); // tells if the Millhoue's wheel
is animating
```

~IsDead(NPC);

Parameter(s): 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(_lolo)) { [Iolo is dead]; }
```

~SetAnim(Item, State); *

Parameter(s): 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:

```
startAnimation = ~SetAnim(_Millwheel, _TRUE); // Enables animation
```

~GetTarget(); *

Parameter(s):

Function: Returns the instanced 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);

Parameter(s): 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 the operation is performed successfully; False if an error occurs.

Example:

```
playHarp = ~Sfx(_Harp + 1); // plays second harp sound
```

Ultima VII Routine and Function declarations

~GetMusic();

Parameter(s):

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

Returns: The number of the current music piece.

Example:
```
oldTune = ~GetMusic();
```

~GetAllInRegion(Type); *

Parameter(s): Type is a valid type for a NPC, monster, terrain piece or item.

Function: Searches region for all occurrences of the 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
```

Ultima VII Routine and Function declarations

Routines

SetSpeaker(NPC);

Parameter(s): 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 picture is already loaded, it switches the text output to match the desired NPC.
Returns:
Example:

```
SetSpeaker(_Iolo);
```

CloseSpeaker(NPC);

Parameter(s): 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 speakerXmax of three portraits at a time)
Returns:
Example:

```
CloseSpeaker(_Iolo);
```

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

Parameter(s): Keyword is a list of strings added to the items menu for the user to select
Function: places a word or words on the menu of active key words
Returns:
Example:

```
TurnOn(<< "hi", "bye" >>);
```

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

Parameter(s): Keyword is a list of strings removed from the items menu for the user to select
Function: Removes a word or words from the menu of active key words
Returns:
Example:

```
TurnOff(<< "hi", "bye" >>);
```

SetWorkType(NPC, Mode);

Parameter(s): NPC is a valid number for an NPC
Mode denotes whether the party is in non-combat or combat
Function: Changes the work type of the desired NPC to the specified work type.
Returns:
Example:

```
SetWorkType(_Iolo, _Dead); // [Bob just killed Iolo, how sad]
```

JoinParty(NPC);

Parameters: NPC is a valid number for an NPC
Function: Sets the NPC into party mode.
Returns:
example:

```
JoinParty(_Bob); // [Bob, leads his strong arm to your quest]
```

LeaveParty(NPC);

Parameters: NPC is a valid number for an NPC
Function: Removes the NPC from party mode
Returns:
Example:

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

Ultima VII Routine and Function declarations

ClearKeys();

Parameter(s):

Function: Clears all keywords from the user interface

Returns:

Example:

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

PushKeys();

Parameter(s):

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

Returns:

Example:

PopKeys();

Parameter(s):

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 arc back
```

SetMusic(Music);

Parameter(s): Music is the number of the required song to play

Function: Loads a new tune into the Music Driver

Returns:

Example:

```
playMusic = SetMusic(someRealNeatTune);
```

PostAction(Item, Script);

Parameter(s): Item is a legal number for an Item, NPC, creature or terrain piece.
Script is a string of legal commands separated 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, a0UT, aEND" );
```

Ultima VII Routine and Function declarations

Curated Findings

These were found in other Ultima 7 development and designer docs from the Internet and in Ultima 7 The Complete Edition. The Parameter(s), Function, and Returns are extrapolated from the code example.

Narrate(String);

Parameter(s): String is the narration dialog.
Function: Posts a narration message on screen.
Returns: Displays a message on screen without a portrait and in a different color.
Example:

```
Narrate("Suddenly, Rotiel doubles over in intense pain!");
```

Found: Ultima 7 – Conversation Stylebook, page 3

PCSays(String);

Parameter(s): String is the dialogue for the PC.
Function: Posts the dialog from the PC.
Returns: Brings up an Avatar's portrait if one is not already on screen, and make the following text appear as if it belonged to the PC. If the picture is already loaded, it switches the text output to match the desired PC.
Example:

```
PCSays("What is thy job?");
```

Found: Ultima 7 – Conversation Stylebook, page 6

AllowChange();

Parameter(s):
Function: To add the keywords for changing the subject.
Returns:
Example:

```
PCSays("Who are the people who call thee such names?");
["Just my friends -- Gentrid, Lanni, and Foul Nondo."];
PushKeys();
TurnOn(<<"Gentrid", "Lanni", "Foul Hondo">>);
AlloWChange();
```

Found: Ultima 7 – Conversation Stylebook, page 9

~GetStudent();

Parameter(s):
Function: *estimate* Retrieves the Student that can train.
Returns:
Example:

```
~GetStudent ();
```

Found: Ultima 7 – Training Code Changes, page 1

~CanTrain(<<STATTYPELIST>>, COST, TRAINER, Student);

Parameter(s): STATTYPELIST can be a single or a list of constants for the type of stat(s).
COST is a set training cost.
TRAINER is a constant of the trainer.
Student is the received training.
Function: *estimate* Returns a boolean for true or false.
Returns:
Example:

```
~CanTrain(<<_Dex, _Str>>, 30, _Inamo, Student);
```

Found: Ultima 7 – Training Code Changes, page 1

Ultima VII Routine and Function declarations

~DexTrain(Student, Int);

Parameter(s): Student is the received training.
Int is the number of times training can be done.
Function: *estimate* Allows the student to train based on Int.
Returns:
Example:

```
~DexTrain(Student, 1);
```

Found: Ultima 7 – Training Code Changes, page 1

~PartyNPCs();

Parameter(s):
Function: Gets the list of NPC numbers of the characters in the party.
Returns:
Example:

```
Party = ~PartyNPCs(); // Variable Party now equals the list of the party members
```

Found: Ultima 7 – Conversation Function and Routines, page 1

~NPCToString(NPC);

Parameter(s): NPC is a valid number for an NPC
Function: Converts an NPC number to the string it represents.
Returns:
Example:

```
Dupre = ~NPCToString(_Dupre); // Variable Dupre now equals the string "Dupre"
```

Found: Ultima 7 – Conversation Function and Routines, page 1

~ChangeStat(NPC, Stat, Int);

Parameter(s): NPC is a valid number for an NPC
Stat is a valid number for the stat
Int is a correct number for the stat. Can be + or – 1 to 255.
Function: Allows an NPC's stats to be changed, or returns false if unable to.
Returns:
Example:

```
~ChangeStat(_Iolo, _Str, 1); // Attempt to increase Iolo's strenth by 1.
```

Found: Ultima 7 – Conversation Function and Routines, page 1

~IsNear(ID);

Parameter(s): ID is either an Item or NPC number.
Function: Finds out if a character or item is on screen.
Returns:
Example:

```
~IsNear(_Iolo); // Checks to see if Iolo is on screen
```

Found: Ultima 7 – Conversation Function and Routines, page 1

~NPCStatus(NPC);

Parameter(s): NPC is a valid number for an NPC or Creature.
Function: Checks a character for its status.
Returns: Returns a list of statuses if there are more than one.
Example:

```
condition = ~NPCStatus(_Shamino); // Condition equals the status(es) Shamino has
```

Found: Ultima 7 – Conversation Function and Routines, page 1

Ultima VII Routine and Function declarations

~MoveItem (Source, Quantity, Item, Quality, Destination);

Parameter(s): Source NPC #, _Party (or -1), _Storage (or -2), _Create (or -3)
Quantity Any number (within reason).
Item A number (underscore constants yet to be defined).
Quality 0 to 255 (or -1 for locking current condition).
Destination NPC #, _Party (or -1), _Storage (or -2), _Create (or -3)

Function: Moves, creates, deletes, and changes the quality of items.

Returns:

Example:
```
~MoveItem(_Dupre, 1, _Sword, -1, _Avatar);
```

Found: Ultima 7 – Conversation Function and Routines, page 1

~InInv(Subject, Quantity, Item, Quality);

Parameter(s): Subject is the NPC #, _Party, or -1
Quantity is a valid number to be set as the quantity for the item to *be* checked within reason
Item is a valid Type number (underscore constants yet to be defined)
Quality 0 to 255 (or -1 for locking current condition).

Function: Finds out if an item is anywhere in a character or the Party's inventory(ies).

Returns:

Example:
```
if (~InInv(_Bart, 2, _Key, 15) { // Checks to see if Bart has 2 keys of quality 15
```

Found: Ultima 7 – Conversation Function and Routines, page 2

~SetMode(NPC, Mode);

Parameter(s): NPC is a valid number for an NPC
Mode denotes whether the party is in non-combat or combat

Function: This is used to set an NPC to a particular way through the conversation. You might want a character to break off the conversation and attack the Avatar, or fall asleep, or even possibly die.

Returns:

Example:
```
[Iolo gasps his last, then falls dead at your feet.];
SetMode(_Iolo,_Dead);
```

Found: Ultima 7 – Conversation Function and Routines, page 4

Ultima VII Routine and Function declarations

Key THOUGHTS { Code };

Parameter(s): Code is a series of function calls and code run in sequenial order.

Function: Key THOUGHTS are a new concept. The basic idea is to use thoughts rather than single words. To reduce the change of typographical errors in writing conversations, Key THOUGHTS will be defined as Variables at the start of the conversation. The Variables will then be used just as the Keywords themselves have been used in the past.

Returns:

Legend Idea: **KEYWORD = Key THOUGHT**
JOB = "I wonder what his job is."
NAME = "I wonder what his name is."
MURDER = "Perhaps I should ask him about this murder."

Example:

```
JOB = "I wonder what his job is.";
TurnOn(JOB);
Key JOB
{
        [Code goes here];
}
```

Found: Ultima 7 – Conversation Stylebook, page 10

usableindex #String ID;

Parameter(s): #String is the name of an NPC. # is required.
ID is the index ID reference

Returns: The number of the ID reference

Example:

```
usableindex #Avatar 1024;
usableindex #Iolo 1025;
usableindex #Spark 1026;
usableindex #Shamino 1027;
usableindex #Dupre 1028;
```

Found: Ultima 7 Code – Usable Index and Flags, page 1

constant _String ID;

Parameter(s): _String is the name of the Constant. _ is required.
ID is the index ID reference

Returns: The number of the ID reference

Example:

```
constant _Avatar 1024;
constant _Party 1;
constant _Dex 1;
```

Found: Ultima 7 Code – Usable Index and Flags, page 5

globalflag $String = bool;

Parameter(s): $String is the name of the global flag. $ is required.
bool is a true or false value

Returns: A global flag boolean

Example:

```
globalflag $SeenTetra = False; // flag number 0
globalflag $SeenSphere = False; // flag number 1
globalflag $SeenCube = Falss; // flag number 2
globalflag $TetraGone = False; // flag number 3
globalflag $SphereGone = False; // flag number 4
```

Found: Ultima 7 Code – Usable Index and Flags, page 10

Ultima VII Routine and Function declarations

token _String Int;

Parameter(s): _String is the name of the token. _ is required.
Int is a unique number

Returns: The token number of the string reference

Example:

```
token _aRAISE 114; // 'r'
token _aEXTEND 101; // 'e'
token _aTHRUST 116; // 't'
token _aBEND 98; // 'b'
token _aKNEEL 107; // 'k'
token _aLIE 112; // 'p'
token _aSTEP 115; // 's'
token _aSTAND 110; // 'n'
token _aATTACK 97; // 'a'
```

Found: Ultima 7 Code – Usable Index and Flags, page 18

<Placeholder>

Parameter(s): placeholder is a string.

Function: Inside of a print statement a variable's value may be referred by surrounding the variable name with <>. There may not be any expressions within the <>. See Example for the sample.

Returns: placeholder's variable

Example:

```
Name = "Avatar";
[Hello <Name>!!!];
A = 1;
[a times 3 = <a*3>]; // This is illegal!!!
```

Found: Ultima 7 – AGIL – Use Code Language, page 9

Ultima VII Routine and Function declarations

converse { Code }

Parameter(s): Code is a series of function calls and code run in sequenial order.
Function: The converse loop is designed to make the internal loop of a conversation easy to specify. The converse loop automatically prompts the end-user for an entry with the prompt "You say:." The resultant input is stored in a globally accessible variable and can be compared to expressions using the key command.
Returns:
Example:

```
Usable Joey {  // Joey is some pre-defined NPC
               // (using an NPC means talking.)
        converse {
                key ("name", "job") {          // Compare the input
                                               // to these two strings
                [I'm joey, the idiot.];        // if they compare,
                                               // then say this.
        }
        key ("piss off") {
                [Hey, screw you!];
                break;                         // Terminates the converse
        loop
}
key * {                // The default key matches 'anything
        [I don't know what you're talking about!];
}
YewInformation();      // Calls the YewInformation Routine
WorldInformation();    // Calls the WorldInformation Routine

Routine YewInformation {
        key ("lord", "brit") {
                [Yeah, that Lord British is a real jerk.];
        }
}

Routine WorldInformation {
        key ("lord", "brit") {
                [The town of justice is Yew.];
        }
}
```

Found: Ultima 7 – AGIL – Use Code Language, page 11

Ultima VII Routine and Function declarations

Standard AGIL code examples

```
While boolean_expression { while_true_code; }
```
Found: Ultima 7 – AGIL – Use Code Language, page 10

```
Foreach TempVar in List { foreach_code; }
```
Found: Ultima 7 – AGIL – Use Code Language, page 11

```
Routine RoutineName { routine_code; }
```
Found: Ultima 7 – AGIL – Use Code Language, page 11