Below is a detailed description of the CondActs used in ACE. They are divided into groups according to the subject they deal with, such as flags, objects etc

Several abbreviations are used in the descriptions as follows:

locno is a valid location number, or 252 (not created), 253 (worn), 254 (carried) and 255 (which is converted into the current location of the player)

mesno is a valid message

sysno is a valid system message

flagno is any flag

procno is a valid sub-process number

word is a word of the required type, which is present in the vocabulary, or "_" which ensures no-word, not an anymatch as normal

value is a value from 0-255

Conditions

There are four conditions which deal with testing the location of the player as follows:

AT locno
Succeeds if the current location is the same as locno

NOTAT locno
Succeeds if the current location is different to locno

ATGT locno
Succeeds if the current location number is greater than locno

ATLT locno
Succeeds if the current location is less than locno

There are eight conditions which deal with the current location of an object:

PRESENT objno
Succeeds if Object objno is carried, worn or present at the current location

ABSENT objno
Succeeds if Object objno is not carried, worn or present at the current location

WORN objno
Succeeds if Object objno is worn

NOTWORN objno
Succeeds if Object objno is not worn

CARRIED objno
Succeeds if Object objno is carried

NOTCARR objno
Succeeds if Object objno is not carried

ISAT objno locno+
Succeeds if Object objno is at Location locno

ISNOTAT objno locno+
Succeeds if Object objno is not at Location locno

There are 8 conditions which deal with the value and comparison of flags:

ZERO flagno
Succeeds if Flag flagno is set to zero

NOTZERO flagno
Succeeds if Flag flagno is not set to zero

EQ flagno value
Succeeds if Flag flagno is equal to value

NOTEQ flagno value
Succeeds if Flag flagno is not equal to value

GT flagno value
Succeeds if Flag flagno is greater than value

LT flagno value
Succeeds if Flag flagno is less than value

SAME flagno(1) flagno(2)
Succeeds if Flag flagno(1) has the same value as Flag flagno(2)

NOTSAME flagno(1) flagno(2)
Succeeds if Flag flagno(1) is not the same value as Flag flagno(2)

<u>There are 5 conditions which will check for an extended logical sentence:</u>

ADJECT1 word
Succeeds if the first noun's adjective in the current LS is word

ADVERB word
Succeeds if the adverb in the current LS is word

PREP word
Succeeds if the preposition in the current LS is word

NOUN2 word
Succeeds if the second noun in the current LS is word

ADJECT2 word
Succeeds if the second noun's adjective in the current LS is word

The following condition is for random occurrences. You could use it to provide a chance of a tree falling, or a lightning strike, or a bridge collapsing etc

CHANCE percent
Succeeds if percent is less than, or equal to a random number in the range of 1-11 (inclusive). Thus a CHANCE 50 condition would allow ACE to

look at the next condition only if the random number generated was between 1 and 50, i.e. a 50% chance of success

QUIT
SysMess 12 (Are you sure?) is generated and the input routine called. It will succeed if the player replied with **'Y'**. If not, the actions **NEWTEXT** and **DONE** are performed

<u>There now follow 19 actions to deal with the manipulation of object positions:</u>

GET objno
If Object objno is worn or carried, Sysmess 25 (you already have the _) is printed and actions NEWTEXT and DONE are performed

If Object objno is not at the current location, Sysmess 26 (There isn't one of those here) is printed and actions NEWTEXT and DONE are performed

If the total weight of the objects carried and worn by the player, plus Object objno would exceed the maximum allowed (Flag 52) then Sysmess 43 (The _ weighs too much) is printed and actions NEWTEXT and DONE are performed

If the maximum number of objects is being carried (Flag 1 is greater than or equal to Flag 37), then Sysmess 27 (You can't carry any more) is printed and actions NEWTEXT and DONE are performed. In addition, any current DOALL loop is cancelled

Otherwise the position of Object objno is changed to carried, Flag 1 is incremented and Sysmess 36 (You now have the _) is printed

DROP objno
If Object objno is worn then Sysmess 24 (You can't, you're wearing _) is printed and actions NEWTEXT and DONE are performed

If Object objno is at the current location (but neither worn, nor carried), Sysmess 49 (You don't have the _) is printed and actions NEWTEXT and DONE are performed

If Object objno is not at the current location as all then Sysmess 28 (You don't have one of those) is printed and actions NEWTEXT and DONE are performed

Otherwise the position of Object objno is changed to the current location, Flag 1 is decremented and Sysmess 39 (You've dropped the _) is printed

WEAR objno
If Object objno is at the current location (but not carried or worn) then Sysmess 49 (You don't have the _) is printed and actions NEWTEXT and DONE are performed

If Object objno is already being worn, then Sysmess 29 (You are already wearing the _) is printed and actions NEWTEXT and DONE are performed

If Object objno is not carried then Sysmess 28 (You don't have one of those) is printed and actions NEWTEXT and DONE are performed

If Object objno is not flagged as wearable (see object definition table in start file 3) then Sysmess 40 (You can't wear the _) is printed and actions NEWTEXT and DONE are performed

Otherwise the position of Object objno is changed to worn, Flag 1 is decremented and Sysmess 37 (You are now wearing the _) is printed

REMOVE objno

If Object objno is carried, or at the current location (but not worn), then Sysmess 50 (You're not wearing the _) is printed and actions NEWTEXT and DONE are performed

If Object objno is not at the current location then Sysmess 23 (You're not wearing one of those) is printed and actions NEWTEXT and DONE are performed

If Object objno is not flagged as wearable (see object definition table in start file 3), and therefore also removable, then Sysmess 41 (You can't remove the _) is printed and actions NEWTEXT and DONE are performed

If the maximum number of objects is being carried (Flag 1 is greater than, or equal to, Flag 37) then Sysmess 42 (You can't remove the _ . Your hands are full) is printed and actions NEWTEXT and DONE are performed

Otherwise the position of Object objno is changed to carried. Flag 1 is incremented and Sysmess 38 (You've removed the _) is printed

CREATE objno
The position of Object objno is changed to the current location and Flag 1 is decremented if the object was being carried

DESTROY objno
The position of Object objno is changed to not-created and Flag 1 is decremented if the object was being carried

SWAP objno(1) objno(2)
The positions of the two named objects are exchanged. Flag 1 is not adjusted. The currently referenced object is set to be Object objno(2)

PLACE objno locno
The position of Object objno is changed to Location locno. Flag 1 is decremented if the object was being carried. It is incremented if the object is placed at location 254 (carried)

PUTO locno
The position of the currently referenced object (i.e. that object whose number is given in flag 51) is changed to be Location locno. Flag 54 remains its old location. Flag 1 is decremented if the object was carried. It is incremented if the object is placed at location 254 (carried)

DROPALL
All objects which are carried or worn are created at the current location (i.e. all conveyable objects are dropped) and Flag 1 is set to 0. Note that a DOALL 254 will carry out a true DROPALL, taking care of any special actions included

The next 6 actions are automatic versions of GET, DROP, WEAR, REMOVE, PUTIN and TAKEOUT. They are automatic in that instead of needing to specify the object number, they each convert

Noun(Adjective)1 into the currently referenced object – by searching the object word table. The search is for an object which is at one of several locations in descending order of priority – see individual descriptions. This search against priority allows ACE to know which object is 'implied' if more than one object with the same noun description (when the player has not specified an adjective) exists; at the current location, carried or worn, and in the a container, in the case of TAKEOUT

AUTOG
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – here, carried, worn – i.e. the player is more likely to be trying to GET an object that is at the current location than one that is carried or worn. If an object is found, its number is passed to the GET action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 26 (There isn't one of those here) is printed and the actions NEWTEXT and DONE are performed

AUTOD
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – carried, worn or here – i.e. the player is more likely to be trying to DROP an object that is carried than one that is worn or here. If an object is found, its number is passed to the DROP action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 28 (You don't have one of those) is printed and the actions NEWTEXT and DONE are performed

AUTOW
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – carried, worn or here – i.e. the player is more likely to be trying to WEAR an object that is carried than one that is worn or here. If an object is found, its number is passed to the WEAR action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 28 (You don't have one of those) is printed and the actions NEWTEXT and DONE are performed

AUTOR
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – worn, carried or here –

i.e. the player is more likely to be trying to REMOVE an object that is worn than one that is carried or here. If an object is found, its number is passed to the REMOVE action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 23 (You're not wearing one of those) is printed and the actions NEWTEXT and DONE are performed

AUTOP locno
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – carried, worn or here – i.e. the player is more likely to be trying to PUT a carried object inside another, than one that is worn or here. If an object is found, its number is passed to the PUTIN action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 28 (You don't have one of those) is printed and the actions NEWTEXT and DONE are performed

AUTOT locno
A search for the object number represented by Noun(Adjective)1 is made in object word in order of location priority – in container, carried, worn or here – i.e. the player is more likely to be trying to get an object out of a container which is actually in there, than one that is carried, worn or here. If an object is found, its number is passed to the TAKEOUT action. Otherwise if there is an object in existence anywhere in the game, or if Noun1 was not in the vocabulary, then Sysmess 28 (There isn't one of those in the), a description of Object objno and Sysmess 51 (.) is printed and the actions NEWTEXT and DONE are performed

Note: No check is made by either AUTOP or AUTOT that Object locno is actually present. This must be carried out by you if required

COPYOO objno (1) objno(2)
The position of Object objno(2) is set to be the same as the position of Object objno(1). The currently referenced object is set to be Object objno(2)

There are four actions which allow various parameters of objects to be; placed in flags, set from flags – for comparison or manipulation

COPYOF objno flagno
The position of Object objno is copied into Flag flagno. This could be used to examine the location of an object in a comparison with another flag value, e.g.
COPYOF 1 11
SAME 11 38
could be used to check that object 1 was at the same location as the player, although ISAT 1 255 would be better!

COPYFO flagno locno
The position of Object objno is set to be the contents of Flag flagno. An attempt to copy from a flag containing 255 will result in a runtime error of 'Invalid Argument'

WHATO
A search for the object number represented by Noun(Adjective1) is made in object word, in order of location priority – carried, worn or here. This is because it is assumed that any use of WHATO will be related to carried objects rather than any that are worn or here. If an object is found, its number is placed in flag 51, along with the standard current object parameters in flags 54-57. This allows you to create other auto-actions (see tutorial game for dropping objects while up a tree)

WEIGH objno flagno
The true weight of Object objno is calculated (i.e. if it is a container, any objects inside have their weight added – don't forget that nested containers stop adding their contents after ten levels), and the value is placed in Flag flagno. This will have a maximum value of 255, which will not be exceeded. If Object objno is a container of zero weight, Flagno flagno will be cleared, as objects in zero weight containers also weigh zero

Now, ten actions to manipulate the flags:

SET flagno
Flag flagno is set to 255

CLEAR flagno
Flag flagno is cleared to 0

LET flagno value

Flag flagno is set to value

PLUS flagno value
Flag flagno is increased by value. If the result exceeds 255, the flag is set to 255

MINUS flagno value
Flag flagno is decreased by value. If the result is negative, the flag is set to 0

ADD flagno(1) flagno(2)
Flag flagno(2) has the contents of Flag flagno(1) added to it. If the result exceeds 255, the flag is set to 255

SUB flagno(1) flagno(2)
Flag flagno(2) has the contents of Flag flagno(1) subtracted from it. If the result is negative, the flag is set to 0

COPYFF flagno(1) flagno(2)
The contents of Flag flagno(1) are copied to Flag flagno(2)

RANDOM flagno
Flag flagno is set to a number from the Pseudo-random sequence from 1-100. This could be useful to allow random decisions to be made in a more flexible way than with the CHANCE condition

MOVE flagno
This is a very powerful action designed to manipulate PSIs. It allows the current LS verb to be used to scan the connections table for the location given in Flag flagno. If the verb is found then Flag flagno. is changed to be the location number associated with it, and the next condact is considered. If the verb is not found, or the original location number was invalid, then ACE considers the next entry in the table – if present. Thus you could consider that ACE carries out the following imaginary entries on exit from Process Table 0 if no action has been done:

```
_     _     MOVE 38   #    Attempt to move player
            DESC      #    Describe new location

_     _     LT 33 15  #    Movement word?
```

```
                SYSMESS 7 #       Can't go that way
                DONE

_       _       SYMESS 8   #      Can't do that
```

This feature could be used to provide characters with random movement in valid directions; by setting the LS verb to a random movement word and allowing MOVE to decide if the character can go that way. Note that any special movements which are dealt with in Process Table 0 for the player, must be dealt with separately for a PSI as well

Now three actions to manipulate flags dealing with the player:

GOTO locno
Changes the current location to locno. This effectively sets flag 38 to the value locno

WEIGHT flagno
Calculates the true weight of all objects carried and worn by the player (i.e. any containers will have the weight of their contents added up to a maximum of 255). This value is then placed in Flag flagno. This would be useful to ensure the player was not carrying too much weight to cross a bridge without it collapsing etc

ABILITY value(1) value(2)
This sets Flag 37 (maximum number of objects conveyable) to value(1) and Flag 52 (maximum weight of objects carried or worn at one time) to value(2). No checks are made to ensure that the player is not already carrying more than the maximum. GET and so on, which check the values, will still work correctly and prevent the player carrying any more objects, even if you set the value lower than that which is already carried

Odds and Ends

PRINT flagno
The contents of Flag flagno are printed. This is a very useful action which can be used for printing current scores or other changing functions like how many coins are left after spending some. For example if Flag 100 contained the number of coins (e.g. 8) carried then:

```
SYSMESS 9        # (You have )
PRINT 100        # 8
```

MESSAGE 11 # (gold coins left)
could be used to show the player how many were left

TURNS
Sysmess 17 – 20 (You have taken XX turn(s)) is printed where XX is Flag 31 + 256*Flag 32

SCORE
Symess 21-22 (You have scored XX points) is printed where XX is the contents of Flag 30

CLS
Clears the screen

NEWLINE
Effectively does a carriage return

MESSAGE mesno
Prints Message mesno, then carries out a NEWLINE action

SYSMESS sysno
Prints System Message sysno

MES mesno
This is a message printed without being automatically preceded by a Newline. It is most useful where an addition or alteration is needed to a Location Description

For Example: Perhaps you have a Location Description where there is a shop with a shopkeeper behind the counter. It may be that you do not want the shopkeeper available at all times and so a Flag must be allocated for that purpose. Let us say Flag 100. You can then put a message in the Message Table that says, "The shopkeeper is in the shop." Let us say that it is Message No.10. Now you must remember to leave a single space at the end of the appropriate Location Description. Let us say that the shop is Location 5. You now need to put an entry in Process Table One as follows: -

```
_    _    AT          5     (At Location 5)
          NOTZERO     100   (Flag 100 is not zero)
          MES         10    (Print message 10)
```

```
                NEWLINE         (Newline)
                LISTOBJ         (List available objects)
                DONE            (End of Process)
```

At the beginning of the game Flag 100 will automatically be zero so it will be up to you to select a way of setting flag 100. It can be anything that you choose but, providing Flag 100 is Notzero when you enter the shop, the shopkeeper will be there. Turning Flag 100 back to Zero will remove him from the shop again

The above process can be used for all manner of things other than appearing and disappearing characters. You can use it to describe exits from the location such, "The oak door is closed" or "The oak door is open." Etc The command is very flexible. If you require more than one choice of extra message added to the Location Description then extra entries will be needed in Process Table One. For example, using the same example Location 5 and using Flag 100 to represent a door (i.e.) ZERO = Closed and NOTZERO = Open with Messages 10 and 11 then the entries in Process Table One would be as follows: -

```
_     _     AT        5     (At Location 5)
            ZERO      100   (Flag 100 is Zero)
            MES       10    (Print "The door is closed
            NEWLINE         Newline
            LISTOBJ         (List available objects)
            DONE            (End of Process)

_     _     AT        5
            NOTZERO   100   (Flag 100 is not zero)
            MES       11
            NEWLINE
            LISTOBJ
            DONE
```

LISTOBJ
If any objects are present then Sysmess 1 (You can also see) is printed, followed by a list of all the objects present in the current location. If there are no objects then this message is not printed

LISTAT locno

If any objects are present then they are listed. Otherwise Sysmess 53 (nothing) is printed – note that you will usually have to precede this action with a message along the lines of (in the bag is) or something like that. It would be possible to create an alternative to the INVEN action (see next) by using 253 and 254 as parameters for LISTAT

INVEN
Sysmess 9 (You have) is printed. If no objects are carried or worn, Sysmess 11 (nothing at all) is printed. Otherwise the object text for each object that is carried or worn is printed, then action DONE is performed

DONE
This action jumps to the end of the Process Tables and flags to ACE that an action has been carried out, i.e. no more entries are to be considered. A return will thus be made to the previous calling Process Table or to the start point of any active DOALL loop

SAVE
The standard save routine allowing the player to save their current position

LOAD
The standard load routine allowing the player to reload a previously saved position

ANYKEY
Sysmess 16 (Press the spacebar to continue..) is printed and the game waits until the spacebar is pressed

UNDO (or OOPS)
This command will take the player back one move in the game and can be used 5 times in succession, after which, system message 61 comes into effect which says something to the effect that 'you can't go back any further'. If any score has been accrued on the moves being undone then flag 30 will automatically be decremented by that amount, and any flags reset to their pre-undo condition. The word UNDO (and/or OOPS) will need to be inserted into the vocabulary table as a verb, and the appropriate message added to System Message 61 if not already present. The command should be added into Process Table 0 as:

```
UNDO        _      UNDO
                   DONE
```

UNDOCLR

This command can be used to prevent the use of the UNDO command to allow for points of no return, e.g. if pressing a switch caused you to fall through a trapdoor and you did not want the player to have any way back, you could use the command as shown below:

```
PRESS SWITCH       AT 10
                   MESSAGE 100 (Press switch and fall)
                   PAUSE 5 (wait 5 seconds)
                   UNDOCLR (prevent the UNDO command)
                   GOTO 11 (go to basement location)
                   DESC
```

AGAIN

This command will repeat the last command that was entered on the command line. System Message 62 will need to be added with a message like 'There is nothing to repeat' to trap those occasions, such as the start of a game, where the use of AGAIN would achieve nothing. The word AGAIN will need to be added to your vocabulary table as a verb. It should be entered into Process Table 0 as follows:

```
AGAIN       _      AGAIN
                   DONE
```

PAUSE value

The game is paused to value (seconds)

PROCESS procno

This transfers the attention of ACE to the specified Process Table number. Process Tables form the heart of ACE, providing the main game control:

Process 0	Each entry contains the word values of the Verb and noun for the LS the entry is to deal with, followed by any number of condacts. When the adventure is played if there is an entry in the table which matches the verb and noun1 of the LS entered, then the condacts are performed

Process 1	Is scanned by ACE after a location is described to allow any additional information which forms part of the location description to be displayed
Process 2	Is scanned by ACE every phrase extracted from the player input

Further Process Tables can be added as required. From Process Table 3 onwards, the verb/noun sequence is not needed, but they can be used by the author to provide a note of the function of any entries within the tables

The object text table
This table has an entry for each object, and the text for that object is printed when that object is described

The object attributes table
This table has an entry for each object and contains the start location (not-created by default), the weight, whether that object is a container, whether it is wearable, the object noun and any adjective required

The location text table
This table has an entry for each location, and contains the text which will be printed when a location is described

The message text table
This table contains the text of all the messages the author puts into the adventure

The System Message table
This table contains all the automatic messages used by ACE – these messages can be adapted to suit the author's requirements if needed

The connections table
This table has an entry for each location and each entry should contain the available exits. If there are to be no obvious exits from a location the location entry should still be there but have no exits listed

The vocabulary table

This table contains all the words that the game needs to recognise and should be laid out in sections:
VERBS
NOUNS
ADJECTIVES
PREPOSITIONS
PRONOUNS
CONJUGATIONS
ADVERBS
Plenty of synonyms should be included but they should have the same word value as the first (e.g. if get is 20 then take should be value 20 also)

The next two actions completely exit any Process Table activities:

DESC
Will cancel any DOALL loop, any sub-process calls and make a jump to describe the current location

END
Sysmess 13 (Another try?) is printed and the input routine called. Any DOALL loop and sub-process calls are cancelled

Flags
Most of the lower number of flags have a pre-defined use and should not be used by the game writer (see below). Since the number of flags available is virtually unlimited, it is suggested that the game writer starts using flags from 70 onwards.

Pre-defined flags:

Flag 0	When non-zero, indicates the game is dark (see also Object zero
Flag 1	Holds the quantity of objects carried by the player (but not worn)

The following flags are decremented if non-zero, by ACE:

Flag 2	When a location is described
Flag 3	When a location is described and it's dark
Flag 4	When a location is described, it's dark and object 0 is absent
Flags	5-8Every time frame (see also autodec and autoinc at end)
Flag 9	Every time frame that it's dark
Flag 10	Every time frame that it's dark and object 0 is

	absent
Flag 30	Holds the current score
Flag 31-32	Holds the number of turns the player has taken
Flag 33	Holds the verb for the current logical sentence
Flag 34	Holds the first noun in the current logical sentence
Flag 35	Holds the adjective for the first noun
Flag 36	Holds the adverb for the current logical sentence
Flag 37	Holds the maximum number of objects conveyable (set using the Ability action)
Flag 38	Holds the current location of the player
Flag 43	Holds the preposition in the current logical sentence
Flag 44	Holds the second noun in the current logical sentence
Flag 45	Holds the adjective for the second noun
Flag 46	Holds the current pronoun ('it' usually) noun
Flag 47	Holds the current pronoun ('it' usually) adjective
Flag 51	Holds the last object referenced by GET/DROP/ WEAR/WHATO etc. This is the number of the currently referenced object which will be printed in place of any underlines in the text
Flag 52	Holds the player's strength (maximum weight of objects carried and worn, and set with the Ability action)
Flag 54	Holds the present location of the currently referenced object
Flag 55	Holds the weight of the currently referenced object
Flag 57	Is 128 if the currently referenced object is a container
Flag 57	Is 128 if the currently referenced object is wearable

Flags 5-8 decrease automatically by one, each time frame, if non-zero, but any flag can be set to automatically increase or decrease with the commands AUTODEC and AUTOINC. The following is an example of how you would apply the programming assuming you had an empty bowl that you put some snow in, and it then melted:

In process table 0 (or whatever table you use for the GET command)

```
GET    SNOW AT (Location no)
          PRESENT (Obj no)      #have bowl in inventory
            LET 300 7           #set flag (any one you like) to
                                #The number required
          AUTODEC 300 1         #turn autodec on
          MESSAGE  (mesno)      #tell the player there is snow in
                                #Bowl
```

```
                DONE
```

Then in process 2

```
_   _    EQ 300 2              #snow has melted
          SWAP 27 13            #swap bowl of snow for bowl of water or
                                #use a flag to indicate either snow or
                                #water
          AUTODEC 300 0         #turn autodec off
          CLEAR 300             #clear flag or else keep getting message
                                #that the snow has melted
```

Note that except in very special circumstances, there is no DONE or DESC used in process table 2

FILE EXTENSIONS

The two file extensions you will need to know are .asg and .agd. The extension .asg is given when a person playing an ACE game wishes to save, and they are asked for a file name e.g. save1 and the extension .asg is given which simply means Ace Saved Game. Thus when a person wishes to continue a game from a saved point, they can open the save1.asd file. The .agd extension is the file extension given to denote a compiled game file, and means Ace Game Data. A person wishing to play an ACE game simply opens the file WinAce.exe and navigates to the game file e.g Cave.agd

THE CSS FILE

Each compiled game must include a style.css file (you can find a standard one within the provided start files). With this the author can stipulate the background page colour, font colour, size, type etc. Appendix A includes all the colour variations that can be used within this style.css file.

ADDING SOUNDS/GRAPHICS AND CHANGING FONTS

Still graphics and sounds can be added, plus font styles, sizes and colours can be changed within the game. An example of how to use each has been included in the sample game file 'Lunch In The Park'. It is worth noting that sometimes you will need to use the colour number (as per Appendix A), rather than the colour name

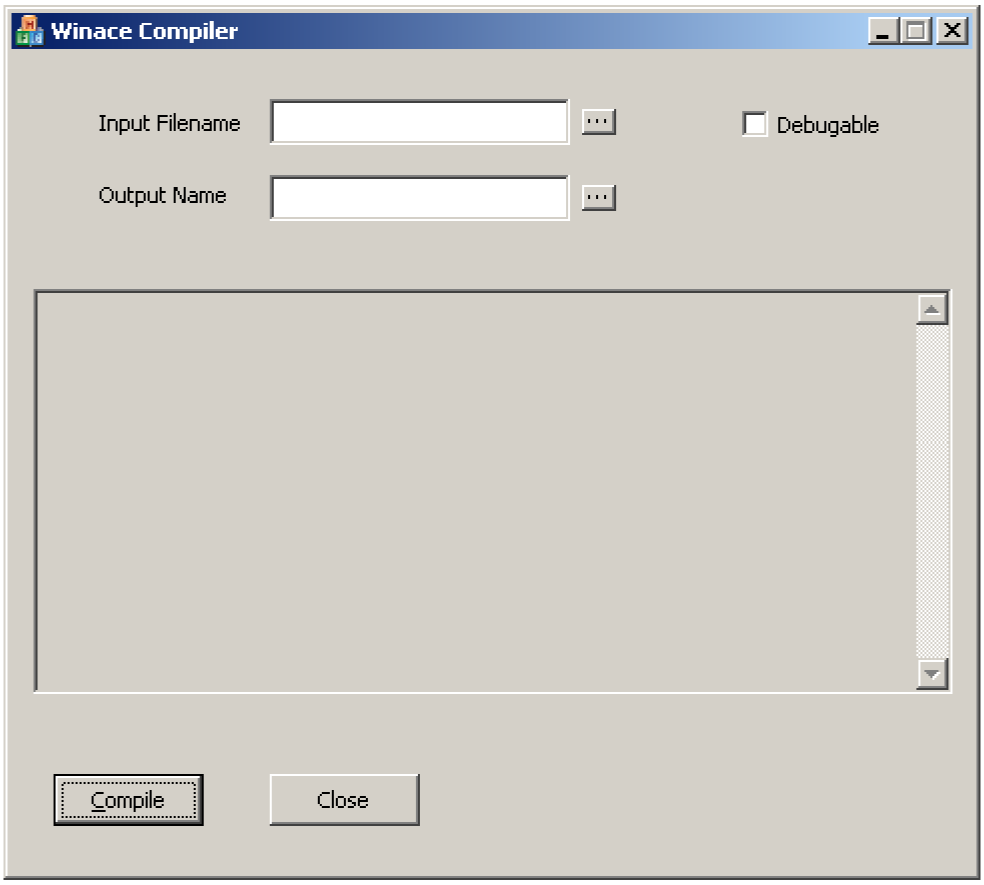
LIMITS

Effectively there are no true limits within ACE but the following has been suggested by the ACE author:

1,000 objects
10,000 locations
5,000 messages of each type
1,000 process tables
1,000 of each word type

THE COMPILER

As you will have discovered from earlier sections of this manual, ACE is produced with a set of text files. For example, let us assume that you are writing a game called CAVE ADVENTURE. It is possible that your text files have been saved as "cave1.txt", "cave2.txt", "cave3.txt" etc. To produce a playable game, you will need to compile the text files into something that the computer can understand. Clicking on the "Compiler.exe" icon will bring up the compiler screen. You will be asked for your Input Filename, an Output name and whether you wish the game to be De-bugable or not. There are three dots to the right of the filename windows and clicking on these will allow you to browse for the file that you need. In our example, in the Input window you must input is the first text file in the series, "cave1.txt". In the Output window you must put the name of your finished game with the suffix .agd. This means Ace Game Data.



Thus in the above example the name in the second window might be "Cave.agd". The small window asking whether the game should be de-bugable or not should be ticked until you have finished the game writing. Once this has been done, click on the "Compile" button to produce a playable game. If you have made any syntax errors in your text files, an "Error" response will occur telling you where the error has occurred. Ticking the "De-bugable" window will enable you to check on flags and manipulate objects until the game has been tested. To do this click on the "View" menu whilst playing the game and then you can select either "Flags" or "Objects" and alter them for testing purposes. This facility is not available if the "De-bugable" window has not been ticked. If you minimise the compiler on your taskbar when you have finished, you will only need to do this once until your programming session is over and you close the compiler. When all of your play testing has been finished, a final compiled game can then be produced with the window unticked for public domain use.

APPENDIX A

CSS Colour Names

Colour Name	Colour number	Colour
AliceBlue	#F0F8FF	⬜
AntiqueWhite	#FAEBD7	⬜
Aqua	#00FFFF	🟦
Aquamarine	#7FFFD4	🟩
Azure	#F0FFFF	⬜
Beige	#F5F5DC	⬜
Bisque	#FFE4C4	⬜
Black	#000000	⬛
BlanchedAlmond	#FFEBCD	⬜
Blue	#0000FF	🟦
BlueViolet	#8A2BE2	🟪
Brown	#A52A2A	🟫
BurlyWood	#DEB887	🟫
CadetBlue	#5F9EA0	🟦
Chartreuse	#7FFF00	🟩
Chocolate	#D2691E	🟧
Coral	#FF7F50	🟧
CornflowerBlue	#6495ED	🟦
Cornsilk	#FFF8DC	⬜
Crimson	#DC143C	🟥
Cyan	#00FFFF	🟦
DarkBlue	#00008B	🟦
DarkCyan	#008B8B	🟦
DarkGoldenRod	#B8860B	🟨
DarkGray	#A9A9A9	⬛
DarkGreen	#006400	🟩
DarkKhaki	#BDB76B	🟨
DarkMagenta	#8B008B	🟪
DarkOliveGreen	#556B2F	🟩
Darkorange	#FF8C00	🟧
DarkOrchid	#9932CC	🟪
DarkRed	#8B0000	🟥
DarkSalmon	#E9967A	🟧
DarkSeaGreen	#8FBC8F	🟩
DarkSlateBlue	#483D8B	🟪
DarkSlateGray	#2F4F4F	⬛
DarkTurquoise	#00CED1	🟦
DarkViolet	#9400D3	🟪
DeepPink	#FF1493	🟥
DeepSkyBlue	#00BFFF	🟦
DimGray	#696969	⬛
DodgerBlue	#1E90FF	🟦
Feldspar	#D19275	🟫
FireBrick	#B22222	🟥
FloralWhite	#FFFAF0	⬜
ForestGreen	#228B22	🟩
Fuchsia	#FF00FF	🟪
Gainsboro	#DCDCDC	⬜
GhostWhite	#F8F8FF	⬜

Gold	#FFD700	🟨
GoldenRod	#DAA520	🟨
Gray	#808080	⬛
Green	#008000	🟩
GreenYellow	#ADFF2F	🟩
HoneyDew	#F0FFF0	⬜
HotPink	#FF69B4	🟪
IndianRed	#CD5C5C	🟥
Indigo	#4B0082	🟪
Ivory	#FFFFF0	⬜
Khaki	#F0E68C	🟨
Lavender	#E6E6FA	⬜
LavenderBlush	#FFF0F5	⬜
LawnGreen	#7CFC00	🟩
LemonChiffon	#FFFACD	⬜
LightBlue	#ADD8E6	🟦
LightCoral	#F08080	🟥
LightCyan	#E0FFFF	⬜
LightGoldenRodYellow	#FAFAD2	⬜
LightGrey	#D3D3D3	⬜
LightGreen	#90EE90	🟩
LightPink	#FFB6C1	🟥
LightSalmon	#FFA07A	🟧
LightSeaGreen	#20B2AA	🟩
LightSkyBlue	#87CEFA	🟦
LightSlateBlue	#8470FF	🟪
LightSlateGray	#778899	⬛
LightSteelBlue	#B0C4DE	🟦
LightYellow	#FFFFE0	⬜
Lime	#00FF00	🟩
LimeGreen	#32CD32	🟩
Linen	#FAF0E6	⬜
Magenta	#FF00FF	🟪
Maroon	#800000	🟫
MediumAquaMarine	#66CDAA	🟩
MediumBlue	#0000CD	🟦
MediumOrchid	#BA55D3	🟪
MediumPurple	#9370D8	🟪
MediumSeaGreen	#3CB371	🟩
MediumSlateBlue	#7B68EE	🟪
MediumSpringGreen	#00FA9A	🟩
MediumTurquoise	#48D1CC	🟦
MediumVioletRed	#C71585	🟪
MidnightBlue	#191970	🟦
MintCream	#F5FFFA	⬜
MistyRose	#FFE4E1	⬜
Moccasin	#FFE4B5	🟨
NavajoWhite	#FFDEAD	🟨
Navy	#000080	🟦
OldLace	#FDF5E6	⬜
Olive	#808000	🟫
OliveDrab	#6B8E23	🟩
Orange	#FFA500	🟧
OrangeRed	#FF4500	🟧
Orchid	#DA70D6	🟪

PaleGoldenRod	#EEE8AA	🟨
PaleGreen	#98FB98	🟩
PaleTurquoise	#AFEEEE	🟦
PaleVioletRed	#D87093	🟥
PapayaWhip	#FFEFD5	⬜
PeachPuff	#FFDAB9	🟧
Peru	#CD853F	🟫
Pink	#FFC0CB	🟥
Plum	#DDA0DD	🟪
PowderBlue	#B0E0E6	🟦
Purple	#800080	🟪
Red	#FF0000	🟥
RosyBrown	#BC8F8F	🟫
RoyalBlue	#4169E1	🟦
SaddleBrown	#8B4513	🟫
Salmon	#FA8072	🟧
SandyBrown	#F4A460	🟧
SeaGreen	#2E8B57	🟩
SeaShell	#FFF5EE	⬜
Sienna	#A0522D	🟫
Silver	#C0C0C0	⬜
SkyBlue	#87CEEB	🟦
SlateBlue	#6A5ACD	🟪
SlateGray	#708090	⬛
Snow	#FFFAFA	⬜
SpringGreen	#00FF7F	🟩
SteelBlue	#4682B4	🟦
Tan	#D2B48C	🟫
Teal	#008080	🟦
Thistle	#D8BFD8	🟪
Tomato	#FF6347	🟥
Turquoise	#40E0D0	🟦
Violet	#EE82EE	🟪
VioletRed	#D02090	🟥
Wheat	#F5DEB3	🟨
White	#FFFFFF	⬜
WhiteSmoke	#F5F5F5	⬜
Yellow	#FFFF00	🟨
YellowGreen	#9ACD32	🟩