amigaguide

COLLABORATORS

	TITLE : amigaguide	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

amigaguide

1.1 EAC v1.4B User Manual

1.2 Introduction

introduction
Although EAC is not a proper programming language and can be used only
to create text adventure games, I think due to its syntax and style of
variable handling, common in many languages, such as BASIC, COBOL & C,
I think it is fair to call it a `kind programming language'. EAC
operates by reading scripts which are written using a text editor such
as ED or Transwrite. These scripts are then loaded into the EAC program
its self.
If you have ever written an AmigaGuide before, then you should be able
to pick up the EAC language fairly quickly. EAC is slightly more complex
and confusing than AmigaGuide and its syntax can be a pain in the arse
to get a grip of. BASIC programmers should also be able to create
adventures with EAC also fairly quickly as it has many similarities.

1.3 EAC Programming Rules

EAC programming rules
As with all programming languages, there are certain rules that you
must follow when creating EAC adventure scripts. These are:
* All commands are UPPERCASE and have a colon after them.
* Each command MUST be on a separate line and located at the START
of the line. No other text must included after the command or the
command and its arguments.
* Page names and variable names are case sensitive. i.e. if you
have a page name as `MyHouse', EAC will not find `MyHouse' if you
look for `myhouse'.
* Variable names MUST NOT begin with a number. i.e. Where: Chapter2 is
legal, 2ndChapter is not. Also do not use any colons or comma in
variable names.

1.4 First Steps In EAC

first steps in EAC
When a script is loaded EAC will search for a page name called
`START' - Note that start is in CAPITALS:
PAGE:START
Each page must have an end. this is represented by using the command:

ENDPAGE:

Any adventure text must go in between PAGE:[PageName] and ENDPAGE: Any text left outside any pages will be ignored. This can be an advantage by using this as a kind of reminder function.

AmigaGuide users will recognize that PAGE: and ENDPAGE: is similar to `@NODE and @ENDNODE.'

To go to a page use the command GOTO:[PageName] like this:

GOTO:MyHouse

EAC will then look for a page named `MyHouse', which would be in your script as PAGE:MyHouse [Case sensitive remember!]

To give the adventurer an option or options to go somewhere use the command ADDOPT:[Text]:[PageName] You are limited to six options per page. Use the command like this:

ADDOPT:Go into MyHouse:MyHouse

Unlike AmigaGuide, EAC uses a primitive variable system, which allows you to store information about any items you may have or may not have. To use a variable you must like many programming languages, firstly declare it. To do this use the command DEC:

DEC:[VariableName]

By using this command you can declare a variable name, which can be used later on in the adventure. *** NOTE *** All declarations of variables MUST be done before the start of the adventure script.

Okay, let me give you an example:

DEC:Knife

I have now declared a variable called `Knife' - Case sensitive remember!! - which now equals zero. This could mean that I have not collected it yet. We can also declare a variable and change its value at the same time - like this:

DEC:Knife=1

Knife now has a value of 1 - which could mean I now have it.

To change the value of `Knife' we will have to use the LET command, like this:

LET:knife=2

I have now changed the value of Knife to 2. But have I? - No actually. I deliberately made a mistake in the variable name.

By calling then variable `knife' instead of `Knife', EAC would have thrown a wobbly if I had done this in a real script. So remember - Case sensitive! It should be like this!:

LET:Knife=2

Now our Knife equals 2! Okay, so now we no how the variables work, do we
can use them in the various IF commands.
ADDOPTIF:[VariableName=Value]:[Text]:[PageName]
The command ADDOPTIF: will only add an option IF a variable equals
a set value. i.e. Say we have the Knife which equals 1 we could
use ADDOPTIF: to give to option of stabbing someone (like my
brother!):
ADDOPTIF:Knife=1:Stab the man:StabMan
If Knife=1 then we will be given the option of going to page `StabMan'. No such option will be given if Knife equals anything else.
GOIF:[VaribleName]=[Value]
The command GOIF: is similar to the GOTO: command, however will
only goto a page IF the variable equals a set value (as in ADDOPTIF).
Example:
GOIF:NeedToilet=1:BathRoom
A few other commands
EXEC:[FileName]
The command EXEC: will attempt to execute a program. This is good
for sound FX etc. i.e.
EXEC:RUN >NIL Play16 Sounds:Cheer.snd
These commands will change to About requester in the game:
TITLE:[AdventureName]
AUTHOR:[YourName]
COPYRIGHT:[Copyright Info]
VERSION:[EAC version programmed for]
For more commands look at the EAC commands section.

1.5 EAC command list

EAC command list
ADD:[Variable]:[Value] : Adds a value to the specified variable
name. i.e. ADD:score:1000
ADDOPT:[Text]:[PageName] : Gives an option to goto a different
page.
ADDOPTIF:[Variable]=[Value]:
[Text]:[PageName] : Gives an option to goto a different
page, only if it is equal to the
specified value.
ADDTEXT:[Text] : Adds some text to the lower status

bar. i.e. `ADDTEXT:Score: ’
ADDTEXTIF:[Variable]=
[Value]:[Text] : Adds some text to the lower status
bar, only if it is equal to the
specified value. i.e.
ADDTEXTIF:apple=1:You have the apple!
ADDVAR:[Variable] : Adds a variables value to the lower
status bar.
ADDVARIF:[Variable]= : Adds a variables value to the lower
[Value]:[Text] status bar, only if it is equal to
the specified value.
AUTHOR:[AuthorName] : Changes the Author details in the
About requester.
BLUE: : Sets the pen colour to blue.
BROWN: : Sets the pen colour to brown.
?BUILD: : Builds up ALL the text if you use
the GOTO: command. Automatically
started by EAC. Use the command
NOBUILD: to disable.
BUSY: : Enables the `Busy...’ text in the
lower status bar, if it has been
previously disabled.
CLR: : Scrolls up 18 blank lines, effectively
clearing all text.
CLRTEXT: : Removes all text from the lower status
bar.
COPYRIGHT:[CopyrightInfo] : Changes the copyright details in the
About requester.
DEC:[Variable][=Value][,] : Declares a variable for later use in
the adventure script.
DIV:[Variable]:[Value] : Divides the specified variable by the
specifies value/variable.
ENDPAGE: : Calls an end to a page.
ENDWRITE: : Used to call an end to the WRITEIF: command.
EXEC:[Filename] : Executes a file.
FONT:[Fontname] : Attempts to load the specified font
from the FONTS: directory. i.e.
FONT:Siesta.font
Note that you should not add the full

font path, only use the font name.
GETRESULT:[Variable] : The specified variable will equal the
last button number pressed by the user,
or -1 if no button was pressed.
GOIF:[Variable]=[Value]:
[PageName] : Goes to the specified page name, only
if the variable equals the specified
value.
GOTO:[PageName] : Goes to the specified page name.
GREEN: : Sets the pen colour to green.
LEDOFF: : Turns off the audio filter.
LEDON: : Turns on the audio filter.
LET:[Variable]=[Value] : Change a variables value. Remember to
Declare the variable before using this!
LINK:[AdventureFile] : Loads the specified adventure game.
MUL:[Variable]:[Value] : Multiplies the specified variable by the
specified value/variable value.
NOBUSY: : Disables the text `Busy...' from the
lower status bar.
PAGE:[PageName] : Set's up a new page. Remember that
EAC will look for PAGE:START every time
a script is loaded!
PAUSE: : This command will cause the text
displaying process to pause until the
`More' gadget is pressed. This command
is ideal for pages with lots of lines,
as it gives the user time to read the
text.
PEN:[Pen Colour] : Changes the text colour. Note that
other people playing your adventures
will have different palettes, so I
recommend sticking to the first 7
colours, as many people will have
Magic Workbench installed.
QUIT: : Forces EAC to Quit.
RED: : Sets the pen colour to red.
RND:[VariableName]:[Value] : Causes the specified variable name to
equal a random number between 0 and the
specified number.

SUB:[VariableName]:[Value] : Subtracts a value from the specified
variable.
TITLE:[AdventureTitle] : Changes the Adventure details in the
About requester.
TRACKFADEIN: : Fades in any modules that are playing.
TRACKFADEOUT: : Fades out any modules that are playing.
TRACKFREE: : Clears any loaded tracker modules from
memory.
TRACKLOAD:[Tracker] : Loads a tracker module.
TRACKPLAY: : Plays the loaded tracker module.
TRACKSTOP: : Stops the current module.
VERSION:[EACVersionRequir.] : You should use this command in your
adventures to tell anyone else trying to
play it, what version of EAC is
required.
WHITE: : Sets the pen colour to white.
WINDOW:[Text] : Changes the EAC window title.
WRITEIF:[VariableName]=
[Value] : Only writes text or executes commands
between WRITEIF: and ENDWRITE: if the
specified variable equals the specified value.
YELLOW: : Sets the pen colour to yellow.

1.6 EAC Command Index

EAC command index

COPYRIGHT:

DEC:

DIV:

ENDPAGE:

ENDWRITE:

EXEC:

FONT:

GETRESULT:

GOIF:

GOTO:

GREEN:

LEDOFF:

LEDON:

LET:

LINK:

MUL:

NOBUSY:

PAGE:

PAUSE:

PEN:

QUIT:

RED:

RND:

SUB:

TITLE:

TRACKFADEIN:

TRACKFADEOUT:

TRACKFREE:

TRACKLOAD:

TRACKPLAY:

TRACKSTOP:

VERSION:

WINDOW:

WRITEIF:

YELLOW:

1.7 Page commands

page commands

Simple page tutorial

WRITEIF: and ENDWRITE:

PAGE:[PageName]

Sets up a page for use in your adventures. All pages must have an end,
done by using the command ENDPAGE: . The first page in your adventure
MUST be called START. i.e. PAGE:START

ENDPAGE:

Calls an end to a page.

GOTO:[PageName]

Goes to the specified page name.

GOIF:[VariableName]=[Value]:PageName]

Goes to the specified page name, only if the specified variable equals the
specified value. Example:

GOIF:Chips>=1:Eat a chip

WRITEIF:[VariableName]=[Value]

Only writes the text and executes commands between WRITEIF: and ENDWRITE:
if the specified variable equals the specified value.

ENDWRITE:

Calls an end to WRITEIF:

1.8 Variable commands

variable commands

DEC:[VariableName]

Before a variable can be used in your text adventure, it must firstly be
declared. To do this, use the DEC: command. Note that ALL declarations of
variables must also be done before the beginning of the start page
(PAGE:START). You can also do multiple declarations in one line by using a
comma in the line:

DEC:[VariableName],[VariableName]

To declare a variable and change its value at the same time use an
equal sign and then the value:

DEC:[VariableName]=[Value]

ADD:[VariableName]:[Value/VariableName]

Adds the specified value to the specified variable. example:

ADD:score:1

SUB:[VariableName]:[Value/VariableName]
Subtracts the specified value from the specified variable. example:
SUB:score:50
MUL:[VariableName]:[Value/VariableName]
Multiplies the specified value to the specified variable. example:
MUL:armour:1.6
DIV:[VariableName]:[Value/VariableName]
Divides the specified value from the specified variable. example:
DIV:health:2
LET:[VariableName]=[Value/VariableName]
Causes the specified variable to equal the specified value. example:
LET:armour=100
RND:[VariableName]:[Value]
Causes the specified variable to equal a random number between -1 and the
specified value. example:
RND:RandomNumber:1000
GETRESULT:[VariableName]
EAC records in an internal variable, the last button to be pressed by
the user. That variable can then be passed onto one of your declared
variables. i.e:
GETRESULT:button
the variable `button', will then equal the last button number you pressed,
or -1 if you have not yet pressed a button.

1.9 User options commands

user options commands
ADDOPT:[Text]:[PageName]
ADDOPT: gives the user an option to go to another page. It's up to you
to tell the user where they'll go or what they'll pick up in the [Text]
part of the command. The [PageName] specifies which page EAC should goto
if the user selects that option. example:
ADDOPT:Enter house:House
ADDOPTIF:[VariableName]=[Value/VariableName]:[Text]:[PageName]
Like ADDOPT: , ADDOPTIF: will only give the user the option to go
somewhere or do something if the specified variable equals the specified
value. example:
ADDOPTIF:key=0:Pick up the key:Pick_Key
PAUSE:

If your adventures have allot of text, you can use the PAUSE: command,
which will halt all scrolling text until the user presses the `MORE'
gadget.
GETRESULT:[VariableName]
EAC records in an internal variable, the last button to be pressed by
the user. That variable can then be passed onto one of your declared
variables. i.e:
GETRESULT:button
the variable `button', will then equal the last button number you pressed,
or -1 if you have not yet pressed a button.

1.10 Lower status bar commands

lower status bar commands
CLRTEXT:
Removes all text from the lower status bar
ADDTEXT:[Text]
Adds the specified text to the lower status bar. example:
ADDTEXT:Score:
ADDVAR:[VariableName]
Adds the specified variable to the lower status bar. example:
ADDVAR:My_Score
ADDTEXTIF:[VariableName]=[Value]:[Text]
Only adds the specified text to the lower status bar if the specified
variable equals the specified value. example:
ADDTEXTIF:Pizza>=1:You have a pizza! Can I have some?
ADDVARIF:[VariableName]=[Value]:[VariableName]
Only adds the specified variable value to the lower status bar if the
specified variable equals the specified value.
NOBUSY:
Disables the `Busy... and ILBM...' messages in the lower status bar. Use
the command BUSY: to re-enable.

1.11 About commands

about commands
TITLE:[AdventureName]
Sets the about title
AUTHOR:[AuthorName]

Sets the author details
VERSION:[EAC version]
I strongly recommend you use this command in every adventure you produce.
It's main purpose is for compatibility reasons. In the VERSION: command
specify the EAC version you have programmed your scripts for. This tells
anyone trying to play your adventures that they need the version you
have specified or higher of EAC, to play them.
COPYRIGHT:[Text]
Used to display any copyright info or misc messages etc.
WINDOW:[Text]
Changes the EAC window title to the specified text. Example:
WINDOW:This is the window title!

1.12 Text commands

text commands
RED: , GREEN: , BLUE: and YELLOW: tutorial.
Pen tutorial
PEN:[PenColour]
Changes the text colour. Note that other people playing your adventures
will have different workbench palettes, so I recommend sticking to the
first 7 colours, as many people will have Magic Workbench installed. Of
course, if you don't like the idea of this command, you can always use
the commands RED: , GREEN: , BLUE: and YELLOW:
RED:
causes EAC to find the closest colour to RED in your workbench palette,
and sets the pen colour to it.
GREEN:
causes EAC to find the closest colour to GREEN in your workbench palette,
and sets the pen colour to it.
BLUE:
causes EAC to find the closest colour to BLUE in your workbench palette,
and sets the pen colour to it.
YELLOW:
causes EAC to find the closest colour to YELLOW in your workbench palette,
and sets the pen colour to it.
BROWN:
causes EAC to find the closest colour to BROWN in your workbench palette,
and sets the pen colour to it.

WHITE:
causes EAC to find the closest colour to WHITE in your workbench palette,
and sets the pen colour to it.
PURPLE:
causes EAC to find the closest colour to PURPLE in your workbench palette,
and sets the pen colour to it.
CLR:
Clears all text and graphics from the window.
PAUSE:
If your adventures have allot of text, you can use the PAUSE: command,
which will halt all scrolling text until the user presses the `MORE'
gadget.
FONT:[FontName]
Attempts to load the specified font from the FONTS: directory. Note that
you should not add the full font path, only use the font name. Example:
FONT:Topaz.font

1.13 General commands

general commands
EXEC:[ProgramName]
Executes the specified program name. example:
EXEC:EAC:Tools/Play16 Sounds:Cheer.snd
FONT:[FontName]
Attempts to load the specified font from the FONTS: directory. Note that
you should not add the full font path, only use the font name. Example:
FONT:Topaz.font
NOBUSY:
Disables the `Busy... and ILBM...' messages in the lower status bar. Use
the command BUSY: to re-enable.
BUSY:
Enables the `Busy... and ILBM...' messages in the lower status bar. Use
NOBUSY: to disable.
BUILD:
Builds up ALL the text if you use the GOTO: command. Automatically
started by EAC. Use the command NOBUILD: to disable.
QUIT:
Forces EAC to shut down, however you should not really use it.

1.14 Tracker commands

tracker commands

Tracker tutorial.

Audio filter tutorial.

TRACKFADEIN:

Fades in any modules that are playing.

TRACKFADEOUT:

Fades out any modules that are playing.

TRACKFREE:

Clears any loaded tracker modules from memory.

TRACKLOAD:[TrackerModule]

Loads a tracker module.

TRACKPLAY:

Plays the loaded tracker module.

TRACKSTOP:

Stops the current playing module.

LEDOFF:

Turns off the audio filter.

LEDON:

Turns on the audio filter.

1.15 Important variable information

important variable information

EAC uses a two byte variable system, meaning that EAC can only handle variables ranging from -32767 to +32767. If you go over or below these limits, EAC my give incorrect results or even crash! In the next version of EAC I intent to sort this problem out by improving the internal debugger.

1.16 EAC Tutorials Index

EAC tutorials index

Tutorial_01 : `Hello world' example.

Tutorial_02 : How to use the ADDOPT: command.

Tutorial_03 : How to change the About info.

Tutorial_04 : How to use the PAUSE: command.

Tutorial_05 : Example adventure showing you how to place

items such as keys, which then allow you to open
doors etc.
Tutorial_06 : How to use the lower status bar for displaying
information such as keys, weapons, money etc.
Tutorial_07 : Adding and Subtracting from variables.
Tutorial_08 : Audio filter example
Tutorial_09 : Module loading/playing/stoping examples
Tutorial_10 : How to use the PEN: command
Tutorial_11 : How to use the RND: , WRITEIF: and ENDWRITE commands.
Tutorial_12 : Using the colour commands.
Tutorial_13 : How to import ILBM pictures into your adventures.
TutoriaL_14 : Using the GETRESULT: command.

1.17 If equal to

IF equal to...
In the given examples of GOIF: and various other IF: commands, I have
stated that the IF part is [VariableName]=[Value], however you ARE allowed
to do more that that. i.e. it is legal to do [VariableName]<>[Value] or
[VariableName]>=[Value]. It is even possible to do [VariableName]=[VariableName]
Firstly, what those symbols mean:
.-------.---.
Symbol	Meaning
=	Equal to.
<	Lower than.
>	Higher than.
<=	Lower OR equal to. You can also do ` =< ’.
=>	Higher OR equal to. You can also do ` >= ’.
<>	Higher OR lower than. You can also do ` >< ’
`-------'---’
Examples of what you can do with ’em:
Example one:
ADDOPTIF:[VariableName]=>[Value]
ADDOPTIF:Apples=>3:You have more than two apples!:Eat_One
Example two:
ADDOPTIF:[VariableName]>[VariableName]
ADDOPTIF:Apples>Oranges:You have more apples than oranges!:Eat_One
AND extensions to all IF commands
A new feature in EAC v1.2 is an `AND’ extension to the `IF’ commands, i.e.
it is now legal to do:
ADDOPTIF:Apples=3 AND Oranges=3:Eat An Apple:Eat_Apple

1.18 Encryptor v1.5

encryptor

Encryptor converts normal EAC scripts into an encrypted binary file format called EAC-2. There are three advantages of using Encryptor. The first is that your code is very secure, as it can't be viewed or modified because it's all encrypted, making it very hard for people to cheat whilst playing your adventures. Secondly, EAC-2 files load around 40% faster into EAC than normal scripts do, due to the fact that all of the jump tables are contained within the code, this does however, make your code a tiny bit bigger. Finally, Encryptor now offers a new advanced encryption format, which allows you to include ILBM's and tracker modules into the code itself, making external data files a thing of the past!

why include files?

Including all of your pictures and modules into your code means that your adventure becomes just one code, and you no longer have to worry about external files.

is encryptor a compiler?

No. Encryptor only scrambles the data so that no one except EAC can read it. Encryptor will not make executable files.

disclaimer

This program should only be used by advanced EAC coders. Make sure you keep a copy of your source code when converting your adventure scripts!

shareware

Please note that you will only be able to use the Encryptor program if you have registered EAC. Please read the shareware notice in the EAC.guide for information about registering this software.

encryptor v1.5 credits

Design & Coding : Matt Briggs

Ideas and comments : Andrew Fitzgerald

Roy Krister Ellingsen

1.19 Problems you may encounter

problems you may encounter

Problem:

EAC states that it cannot find a page when it is clearly there.

Solutions:

1) Have you typed the page name correctly? Remember that page names ARE

case sensitive!
2) Make sure that there are no `hidden' characters in the page name or
PAGE: command. This can often be the case with both page names and
variable names.
3) Delete the entire line and re-type it.

1.20 Further EAC development plans

further EAC development plans
Okay, lets get one thing straight. Any further developments of EAC will NOT
be released unless I get any kind of response from this version. So if you
like this program please contact me and say that you do! So at least I know
I am on to a good thing. If you don't, then I'm sorry, but EAC will most
likely die forever, and I will do something else :(
coming soon... (maybe)
FONT : Hopefully I will add BOLD, ITALIC and UNDERLINE
TEXT GADGET : To allow you to speak to people.
`OR' : `OR' extensions to all the IF commands.
DEBUGGER : A much better internal debugger system.
TONS MORE
COMMANDS AND
EXTRAS : Lots more commands and extensions will be added.
your ideas please!
If you have any ideas or suggestions, then please contact me!!!

1.21 Contacting ENGiNE NiNE DESiGN

contacting ENGiNE NiNE DESiGN
ENGiNE NiNE DESiGN
Matthew Briggs
50 Thicket Drive
Maltby
Rotherham
South Yorkshire
S66 7LB
U.K.
E-Mail: engine9@onlineamiga.demon.co.uk