```
***************** THE MARVELOUS WORLD OF EAMON DELUXE 5.0 ********************
*                                                                            *
*                  EAMON DELUXE ADVENTURE DESIGN MANUAL                      *
*                           Second Edition                                   *
*                                                                            *
************ ALL NON-COMMERCIAL DISTRIBUTION IS ENCOURAGED *******************
```

NOTE: THIS MANUAL IS NOT YET FINISHED!

Because of the large amount of work involved in the Eamon Deluxe 5.0 upgrade, there has not yet been time to create and publish proper gaming manuals. This incomplete document is being released early so that potential Eamon Deluxe adventure authors can have some updated reference material to work with until the Design Manual is properly finished. As you read, please be aware that areas with obviously missing material indicate a placeholder for sections that have not been properly updated yet.

Second Edition by Frank Black with some text by Donald Brown.
Contact: eamondeluxe (at) gmail.com

First Edition published March, 2002. First Edition Final Revision published November, 2007. Second Edition published July, 2012

This manual has been written for those stalwart people who enjoy adventure games and want to design their own under the Eamon Deluxe 5.0 system. It goes without saying that it is great to play Eamon adventures, but the real fun lies in designing your own.

It is assumed that new authors are already familiar with the Eamon Deluxe gaming system as well as the information included in the Second Edition Player's Manual. If special events (beyond those already programmed into Eamon Deluxe) are desired then the author will also need to know (or learn) a little bit about programming in BASIC. Without further ado, lets get on with (what Don Brown called):

HOOKING UP WITH EAMON
or
SENDING ADVENTURERS TO THEIR DEATH FOR FUN AND PROFIT!

Contents

1. EAMON DELUXE 5.0 ADVENTURE DESIGN DIFFERENCES

There are a few notable differences between Eamon Deluxe 5.0 and previous versions when it comes to creating a new adventure. The design and data entry stages are similar but there have been changes regarding adding special programming. More options are now available to the designer and some game data is handled a bit differently. The following list of changes is useful to both experienced and first-time authors and should be read by both.

The base adventure programs (INTRO.BAS and MAINPGM.BAS) have been completely rewritten and are very different from older versions. Only Eamon Deluxe 5.0 specific reference material will apply when creating a new adventure and any manuals or charts from previous versions should be disregarded.

The new "VI Mode", when activated, uses redirection methods to offer alternate menus, programs and descriptions of visual effects for blind and vision impaired gamers who use screen readers or other alternate access software. VI Mode also has other, more subtle differences which make it easier for these special access programs to work properly. If you use the input/output routines already coded into the base adventure programs and don't have any special graphic or sound effects to add then you won't have to worry about VI Mode compatibility at all. If your design includes such things then see section 8, Notes on VI Mode Programming, for more information.

Previous versions of Eamon Deluxe were only available for MS-DOS or 32 bit versions of Windows and were installed to the directory C:\EAMONDX. Eamon Deluxe 5.0 is available cross-platform and installs to different locations depending upon the OS. Windows installs are now located at C:\EDX\C\EAMONDX. OS X versions install into the Applications folder as "Eamon Deluxe 5.0". Linux versions install into /usr/lib/EDX/, depend upon DOSBox also being installed and are launched with the command: dosbox -conf "/usr/lib/EDX/Xeamondx.conf"

The MAINPGM 5.0 is more sophisticated than previous versions in the way that it handles monsters and artifact types. These improvements are listed throughout the Design Manual and won't be repeated here. Internally, it runs as close to Donald Brown's original design as possible and is more authentic in that aspect than any other version of Eamon. However much of this is invisible and involves formulas and calculations which occur "behind the scenes" as the programs run.

1.1 CHARACTER RECORD DATA

Between adventures, all of the characters that the Eamon Deluxe system knows are stored in the following files in the EAMONDX\MAIN directory:

NUMPLRS.DAT: Holds the number of characters on record.
PLAYERS.DAT: The actual character data. Random access file (length of 200).

Data held in a character record: First a string (length=40) containing the character's name. Next, a series of INTEGER values: current Status (1=OK, 2=Dead, 3=On adventure); Hardiness; Agility; Charisma; four Spell Abilities (Blast, Heal, Speed, and Power); five Weapon Abilities (Axe, Bow, Club, Spear, and Sword); Armor Expertise; and Sex (0=Male, 1=Female). Then follow two LONG INTEGER values: Gold carried and gold in the Bank. Then an INTEGER value: Armor Class (Leather=2, Chain=4 Plate=6, with +1 added if a shield is carried). Then four strings (length=20) containing weapon names. And finally (in INTEGER values) the complexity, type, dice, and sides per die of the four weapons. If the character does not have four weapons, then any carried weapons will come first and all other weapon names will either be "NONE" or a null string ("") and the values will be all be zero.

1.2 WHAT HAPPENS WHEN A PLAYER GOES ON AN ADVENTURE

When a character leaves the Main Hall for an adventure, their status is changed to 3 in the PLAYERS.DAT file and character data is written into a 1-record file called "NEW-MEAT" in the adventure sub directory. NEW-MEAT uses the same data format as PLAYERS.DAT except the Status value in NEW-MEAT is used to hold the character record number in PLAYERS.DAT file so the character data can be updated and returned properly when the adventure is over.

After writing to PLAYERS.DAT and NEW-MEAT, the working directory is change to that of the adventure and control is handed to the program INTRO.BAS. INTRO is normally used to display the adventure's opening story and as well as any other options that the author chooses to make available. This program will be referred to as "INTRO" throughout this manual. INTRO, in turn, will end up handing control to the main adventure program (named MAINPGM.BAS by default).

When an adventure is over (or the QUIT HALL command is used), control is returned to the Main Hall. If the adventure ended in the character's death, then their Status is changed to 2 and the Eamon Deluxe Main Menu is run. Otherwise the character is placed in the Main Hall with their new items, updated abilities, etc. and a status of 1.

Note: If you are playing with a character from the Test An Adventure option, then no character updates are made and the Adventure Design Menu is run upon exiting the adventure. The Test An Adventure option uses a negative value as the character record number in NEW-MEAT which so that MAINPGM knows it is a test character.

1.3 CONTENTS OF THE NAME.DAT FILE

Every adventure has a NAME.DAT file which is a sequential text file that holds all the information about the adventure and its database. It contains the following values: Number rooms, artifacts, effects, and monsters; followed by the adventure name, number of directions (6 or 10) the player can move in, the revision number of the Eamon Deluxe System that created it, and the number of adventures stored in the database (always 1 for new adventures).

If the number of adventures is greater than 1, then more data will follow: The name of each adventure followed by that adventure's number rooms, artifacts, effects, and monsters. This is followed by a pointer to the starting record of each data type in the database files and, finally, the number of directions for that adventure.

2. THE ADVENTURE DESIGN PROGRAMS

Start a New Design: This initializes all the proper files for a new adventure into an available design directory and copies the MAINPGM and INTRO and programs. It also copies an executable file called INSTALL.EXE which can be run independently to add a copy of the new adventure to the Main Adventure list when needed. Note: The version of INSTALL.EXE included with Eamon Deluxe 5.0 is backwards compatible with earlier versions of Eamon Deluxe and will install to them if the 5.0 directory is not found.

Rename an Adventure or Design: Self explanatory.

Edit a Design: This is how you can enter/edit your adventure data. Note: There is also a utility that lets advanced users create Adventure Source Code using a plain text file and simple editor such as Notepad for data entry. Email eamondeluxe@gmail.com for more info about that data entry method.

Test a Design: You can select four characters of each gender, with skill levels ranging from weak to ridiculously strong, for use in testing your designs. The strong characters are useful for running through a design without worrying about getting killed and the weak and "average" characters help you adjust the difficulty of your adventure. For a standard adventure, the medium character should be able to survive almost all of the time if played carefully.

Edit Hint Files: Eamon Deluxe allows you to create your own on-line help files which can be called up during the game. These files aren't limited to hints or help only and can contain notes, pranks (like fake hints), etc..

List a Design: Can display the entire contents of an adventure database in a neat and helpful order.

Map Room Connections: Displays quick references of room connections for mapping or other purposes.

Remove a Design: Permanently delete all adventure data and programs and reset an adventure design slot to "empty".

2.1 STARTING A NEW ADVENTURE

[Note: This section needs updating.]

Select Start a New Design from the Adventure Design Menu. Enter the name of your adventure and the number of directions for room connections. Six direction adventures are the preferred standard because they are easier to map and less of a hassle, but the option of making a ten direction adventure is available for those who wish to use it. In the Eamon Deluxe database system -all- adventures are stored in the ten direction file format. In a six direction adventure, the design programs simply ignore the last four connections (NE/NW/SE/SW). MAINPGM will ignore them as well and will also remove their respective commands during initialization.

Once your adventure has been initialized, it will either be in EAMONDX\DGN-1 or EAMONDX\DGN-2 depending upon which design slot you chose. Note that adventures in the design slots can only be played with characters from the Test An Adventure option.

2.2 USING THE ADVENTURE EDITOR

[Note: This section needs updating.]

For each of the four things you can enter (rooms, artifacts, effects and monsters), you can either add a new one to the list, edit one already there, or chose from miscellaneous other options such as deleting or copying.

Eamon Deluxe puts no limits on the amount of any type of data. However, you should never exceed 1,000 of any type because it could cause a conflict with the special room codes and other data fields used by different artifact types.

2.2.1 ENTERING TEXT

[Note: This section needs updating.]

Adding and editing is very similar for all four data types. Every time you enter more than a single key, the entry is done through a special input routine. The following keys perform special functions during entry:

Escape: Accepts all of the text, both before and after the cursor.
Enter: Accepts all of the text from the beginning to the cursor.
Backspace/Delete: Deletes a character.
Tab: Inserts a space.
Left/right/up/down arrows: Move cursor through the text.

The editor has default data field values set up for rooms, artifacts and monsters. This makes it faster to key in data when you want most of the fields to be default. It also displays helpful messages and charts relating to the type of data at hand during entry.

NOTES: Do not begin artifact names with a number; use "eight-inch knife" instead of "8-inch knife". Also, as opposed to other versions of Eamon, -any- text characters (including quotation marks, colons, etc.) may be used in descriptions and names.

When entering -all- descriptions, let text wrap around the screen, DO NOT ADD EXTRA spaces, text will be separated and printed properly by MAINPGM and other programs.

2.2.2 CHAINING OR LINKING TEXT DESCRIPTIONS

[Note: This section needs updating.]

Special programming has been installed for cases where entering a description requires more than the default 255 character limit of the DSC files. Simply enter as much as will fit (and neatly wrap around the screen), then add an asterisk (*) and a three digit number that points to an Effect (see Section 6) where the text is to continue. For example, if you create a monster with the description "You see a rat.*001". The MAINPGM will print "You see a rat.", print a blank line (making a paragraph break), then print the text stored in Effect number one.

If you use two asterisks ("You see a rat.**001"), a paragraph break WILL NOT be printed in between the text and the Effect, making one large paragraph. You can chain an unlimited amount of text with these simple features.

NOTE: For the above process to work, You -must- use three digits. "*1" and "*11" will be ignored (MAINPGM.BAS checks the 4th to last character for an asterisk, then checks for another preceding it), Effect 1 is "001".

3. ENTERING ROOMS

[Note: This section needs updating.]

All "rooms" in an adventure are stored as 14 pieces of data in the random access files ROOMS.DSC (length=255, long description of room) and ROOMS.DAT (length=101, room name, 10 direction values (N/S/E/W/U/D/NE/NW/SE/SW), and the level of natural light in the room (see below).

NOTE: See section 2.2.2 for important notes on entering descriptions.

After the name and description, you will be asked for a room connection value for moving in each direction. Special codes are programmed into MAINPGM:

0: You cannot move in that direction.

1-[number of rooms]: There is an open connection to that room number.

-999: Exit the adventure alive. (Ride off into the sunset.)

[Artifact number] + 1000: Code for a door (1001 being artifact number 1, 1002 = artifact 2, etc.).

Negative numbers (except -999) are reserved for whatever special code you want to program into the MAINPGM. (See Section 7.)

4. ENTERING ARTIFACTS

[Note: This section needs updating.]

An 'Artifact' (with the exception of captives and disguised monsters) is any non-living thing that is in the dungeon.

All Artifacts' in an adventure are comprised of 10 pieces of data and stored in the following random access files: ARTIFACT.DSC (LEN=255), long description of artifact; ARTIFACT.DAT (LEN=51), artifact name and its 8 pieces of data (see below)

NOTE: You must have as artifacts, all weapons used by your monsters.

NOTE: See sections 2.2.1-2.2.2 for notes on entering descriptions.

There are thirteen pre-programmed types of artifacts already recognized by the MAINPGM, which automates the most commonly used item types.

NOTE: Artifact names should be entered in lower-case only. Only use capitals for proper nouns (e.g. "bound troll", "chained Princess Jane").

4.1 ARTIFACT VALUE

A standard range of values was developed while converting the Apple II Eamon library to Eamon Deluxe. In the days of Classic Eamon, when anarchy reigned throughout the realm of game design, the same sword could be worth 5 gold pieces in one adventure and 500 in the next! The current Eamon Deluxe standards system was developed by referencing the Classic Eamon game manuals, examining the original set of adventures and Main Hall written by Donald Brown himself, and then averaging the results.

For a detailed list of artifact value examples, please see Appendix 4 of this manual.

4.2 ARTIFACT WEIGHT

The Eamon Deluxe unit of weight is known as the Grond/Dos system. Why? Because that is what Donald Brown named it almost three decades ago. As mentioned in section 4.1 above, a standard range of values has been developed which is found throughout all of Eamon Deluxe. For a detailed list of artifact weight examples, please see Appendix 4 of this manual.

Artifacts with a special weight value of either -999 or 999 can never be picked up or moved under any normal circumstances. The difference between the two values is that -999 gives the message "You can't get that", while 999 gives the message "Don't be absurd!" Negative numbers can be used for artifacts which allow you to carry more weight than usual (magic bags, etc.).

4.3 ARTIFACT LOCATIONS

Artifacts normally have room numbers ranging from 1 to the number of rooms in the dungeon, but there are additional codes recognized by the MAINPGM.

-1	Carried by player
-(Monster# - 1)	Carried by Monster #
-999	Worn by player
-(Monster# -1000)	Worn by Monster #
Container Artifact # + 1000	Inside a container artifact
Room # + 2000	'Embedded' in a room description

(If you want monster #1 to have it, it is -2. Monster #2 is -3, etc.. If you want monster #1 to wear it, it is -1001.)

4.3.1 EMBEDDED ARTIFACTS

'Embedded' artifacts are those that are not listed in "You see [artifact]" loop initially, but can be acted upon by the player. For example, you could describe a room as containing a statue, and embed a statue artifact. The statue will not be listed as being in the room until it is examined or acted upon by the player.

NOTE: An Embedded artifact MUST be mentioned in another description or it will never be found by the player who must use an artifact manipulation command to cause it to actually 'appear' in the room. If the name of the artifact is a little too revealing (such as 'a secret door to the north') then you can use the Synonym routine to let the player look for something else (see section 7.1 for info on using synonyms).

4.4 ARTIFACT TYPES

There are 13 types of recognized artifacts. You are also allowed to enter a value of 14, 15, or 16 for your own special types. All USER types are treated as type 1 by the MAINPGM unless you add your own code. Creating new artifact types is very uncommon as most any type you'll need is already pre-programmed and it is strongly suggested that you stay within the usual types. The types are as follows:

Type	Format	Type	Format
0. Gold	0	9. Edible	4
1. Treasure	0	10. Bound Monster	7
2. Weapon	1	11. Wearable	8
3. Magic Weapon	1	12. Disguised Monster	9
4. Container	2	13. Dead body	10
5. Lightable	3	14. Undefined, User	0
6. Drinkable	4	15. Undefined, User	0
7. Readable	5	16. Undefined, User	0
8. Door/Gate	6		

All of the formats have the same information in fields 1-4: Value, Type, Weight, and Room #. Format #0 is all USER fields, the rest of the format fields 5-8 contain the following.

Format 1	Format 2	Format 3
5-Weapon Odds	5-Key No.	5-Light Counter
6-Weapon Type	6-Open/Closed(1/0)	6-(USER #6)

7-# of Dice	7-# of items inside	7-(USER #7)
8-# of Dice Sides	8-# items it can hold	8-(USER #8)

Format 4	Format 5	Format 6
5-# of Heal Points	5-1st Effect	5-Room Beyond
6-# of Drinks/Bites	6-# of Effects	6-Key #
7-Open/Closed(1/0)	7-Open/Closed(1/0)	7-Open/Closed(0/1)
8-(USER #8)	8-(USER #8)	8-Hidden?

Format 7	Format 8	Format 9
5-Monster #	5-Armor Class	5-Monster #
6-Key #	6-Armor Type	6-1st Effect
7-Guard #	7-(USER #7)	7-# of Effects
8-(USER #8)	8-(USER #8)	8-(USER #8)

Format 10
5-'Takeable' status*
6-(USER #6)
7-(USER #7)
8-(USER #8)

* -- 0=Can't be taken, 1=Can be. Should ALWAYS be 0 unless the body serves some sort of special function (such as collecting rewards, etc.).

4.4.1 ARTIFACT TYPES: GOLD, TREASURE, AND USER

Gold: Has a set value, like gold pieces.

Treasure: Has a value which will vary depending upon the character's charisma. Base dungeon programs version 4.5 and later won't let the character weasel by with more then 5% greater than the set value, no matter what their charisma.

4.4.2 ARTIFACT TYPES: WEAPONS AND MAGIC WEAPONS

Weapon's type values are 1=Axe, 2=Bow, 3=Club, 4=Spear, 5=Sword. Make all guns and such type 5 (bows), all weapons that the player must lunge and jab with (such as knives and daggers) should be type 4 (spears).

Number of dice and dice sides are used to compute the amount of damage the weapon does in combat. Example: If number of dice = 1 and number of sides = 8 then it will do 1D8 of damage, or from 1 to 8 points of damage. If the two numbers are 12 and 1 (12D1) it will always do 12 points of damage.

Magic Weapons are magic within the current adventure only. If taken back to the Main Hall, it reverts to an ordinary weapon (type 2). These cannot break in combat which is the extent of their magical power. If you want other magic, you'll have to program it (such as "TrollsFire" in the Beginner's Cave). The MAINPGM classes any weapon brought by the player that has a maximum hit potential (#dice x #sides) of 25 or greater as being magical automatically.

4.4.3 ARTIFACT TYPES: CHESTS AND OTHER CONTAINERS

These can contain other artifacts. Artifacts inside are any with a room number of 1000 + the number of the container. The player can also put items inside and remove items from the container if you allow it. If it has a Key value than it is locked and the artifact specified must be used to open it. If the key # is 0 then it is unlocked or doesn't need a key. If a container is locked and has no key (the player must smash it open) code the Open/Closed field with the number of damage hits it must take + 1000. Recap: Closed=0, Open=1, 1000+=Smash it to pieces. If it is locked, has no key, and can't be forced open, code the open/closed field as -1.

An artifact can't be PUT inside a container if its weight is greater than the weight of the container. If the weight of the artifact being PUT plus the weight of the other artifacts inside the container is greater than the weight of the container than the player gets the message, "It's full." If you don't want the player to PUT anything in it at all, code field 8 as 0.

4.4.4 ARTIFACT TYPES: LIGHT SOURCES (LANTERNS, TORCHES, ETC.)

This can be any light source used to illuminate dark rooms. The counter is decremented with each turn (including input mistakes) that the player takes and w
it reaches zero it is extinguished and may not be relit. A if you set the counter
value to -1, MAINPGM will ignore the counter and let it stay lit infinitely.

4.4.5 ARTIFACT TYPES: DRINKABLE AND EATABLE

These can do 1 of 3 things: Heal the user by the 'heal amount' specified, hurt the player by having a negative heal value, or do nothing at all. Drinkable and eatable artifacts may only be used the amount of times specified before they are exhausted.

These 2 types are treated as the same, just worded differently. The only exception is that edible artifacts disappear once the player uses up the number of bites. Note: Don't forget to code food as being 'open'.

4.4.6 ARTIFACT TYPES: READABLE

These can be read by the player. The MAINPGM will automatically read the text from a record in the EFFECT.DSC file that is specified by field 5. You MUST add these Effects, of course. Field 6 is included for compatibility with old Eamon systems, but isn't necessary in Eamon Deluxe because you can chain all text with an asterisk (*) or two (**). See section 2.2.2 on linking text.

NOTE: If you do use field 6 for sequential text, you must make sure the Effects are in the proper order.

NOTE: Even if you link text via the asterisk, field 6 should still be 1.

4.4.7 ARTIFACT TYPES: DOORS

The number of these artifacts + 1000 is put in the room data as the connecting room (see section 3 on ROOMS). Field five tells the room where the player goes when passing through. If field 6 is anything but 0, then the door is locked and the player must have the specified key to open it. If field 7 is zero then the door is unlocked and may be used. If field 7 is greater than 1000 then the door can be broken open. If field 6 is -1 then the door can never be opened via the OPEN command (you must add your own special code).

If you want the door to be hidden or secret, make it an embedded artifact (see section 4.3.1 on EMBEDDED ARTIFACTS) and code field 8 as 1. If a door is hidden then the player will get the message, "You can't go that way!" until the door is discovered. The door will also be ignored if it isn't in the room (visibly or embedded) yet.

4.4.8 ARTIFACT TYPES: BOUND MONSTERS

These are used to simulate a captive. Field 5 is the actual monster to be freed. If field 6 contains anything but 0, a key is needed. If field 7 is anything but 0, then it is the monster # of a guard (who should be in the room also). Freeing the bound monster causes the artifact to disappear from the room and the actual monster to take its place.

4.4.9 ARTIFACT TYPES: WEARABLE ARTIFACTS

This includes all artifacts that can be worn by the player. "Clothing Type" is not presently implemented by the base program, but the following protocol is used in Eamon Deluxe and should be adhered to.

Armor Class	Clothing or Armor Type
0. Clothing	0. Armor, Shields, Plain Clothes
1. Shield	1. Overclothes (Coats, Capes, etc.)
2. Leather Armor	2. Shoes, Boots
4. Chain Mail Armor	3. Gloves
6. Plate Mail Armor	4. Hats, Headwear
	5. Jewelry
	6. Undergarments

4.4.10 ARTIFACT TYPES: DISGUISED MONSTERS

These are artifacts that appear to be normal, but are really a monster in disguise. When a player attempts to get or manipulate the object, it disappears from the room and is replaced with the monster specified in field 5. If it's an enemy, it gets one free shot to attack the player. If you want some special message to be printed then add the Effects and code field 6 with the number of the first effect. Field 7 is included for compatibility with old versions of Eamon. See section 4.4.6 for more information on this obsolete field.

4.4.11 ARTIFACT TYPES: DEAD BODIES

As a general rule, dead bodies are nothing but trouble and really have no use at all. They waste memory space, slow down the program and make late data entry tricky. The option to use them, however, is open to the designer and the MAINPGM has code already in it to handle dead bodies which can be activated by setting the DB variable. Here's what to do if you want dead bodies:

Once you have your adventure COMPLETELY designed and ALL of your artifacts and monsters entered, select the "Special functions" option on the adventure editor menu, and then the "Generate bodies" option. Enter "1" for starting monster. A type 13 artifact will then be generated for each monster in order and given the name "(monster name)'s body". Now, at the "Init:" routine in the MAINPGM enter the line "DB = x", where 'x' is the artifact number of the first dead body.

By setting DB (which normally has a value of zero), you activate the dead body routines in the MAINPGM. When a monster dies, their body will be placed in the room. When a player tries to get a body, they'll get the message, "Dead bodies are best if left alone.". And if they attack the body it will disappear and they'll get the message, "You hack the body to bits!".

If you want the player to be able to pick up the dead body, code field 5 of the dead body artifact as 1.

A lot of Classic Eamon adventures used dead bodies and they were generally kept in the Eamon Deluxe conversions to better keep the flavor of the original.

5. ENTERING MONSTERS

[Note: This section needs updating.]

"Monster" is the generic term used for any living thing in an Eamon Deluxe adventure. MAINPGM loads characters as Monster #0 and treats them similar, however monsters can't cast spells, don't have weight-carried restrictions and have maximum armor expertise and weapon abilities by default.

Monster data is stored in two random access files:

```
 MONSTERS.DAT (Length=71)  - Monster name and 13 integer values (see below)
 MONSTERS.DSC (Length=255) - Long description of monster.
```

NOTE: See sections 2.2.1-2.2.2 on entering text.
NOTE: When creating monsters, refer to APPENDIX 4 for data values.

5.1 ENTERING MONSTERS: HARDINESS AND AGILITY

These values are identical to player Hardiness and Agility. Hardiness is the amount of damage points a monster can take before it dies; During combat Agility and armor help determine the odds of hitting enemies as well as dodging their attacks. The more agile the monster, the greater its ability in battle, but the heavier its armor, the more its agility is compromised.

5.1.1 ENTERING MONSTERS: EFFECTS OF MONSTER VALUES DURING COMBAT

During a battle, the odds of an opponent striking another opponent are 50% plus 2 times the difference in agility and armor between the attacker and the defender.

For an example combat round, let's say a monster (the "attacker") with an agility of 20 and an armor value of 2 is attacking a monster (the "defender") with an agility of 15 and an armor value of 6.

First both the attacker and defender have their agility adjusted by subtracting their armor values from their agility. Then the defender's -adjusted- agility is subtracted from the attackers adjusted agility. The result is multiplied by two then added to the basic 50% chance everyone starts with.

Using that formula, our example attacker would have a 68% chance of hitting the example defender.

Or: 20 - 2 = 18 (attackers adjusted agility), 15 - 6 = 9 (defenders adjusted agility), 18 - 9 = 9 (difference in adjusted agility between the attacker and the defender), 9 times 2 = 18 (double the result of their difference), 50 + 18 = 68 (base chance of 50% everyone has, plus difference in adjusted agility).

If their roles are reversed then the odds to hit are 32%. Or: 9 - 18 = -9, -9 times 2 equals -18. -18 + 50 = 32.

If the attacker is also using a weapon then the above odds are further adjusted by the Complexity (or "quality) of that weapon divided by two.

Using our example battle above, if the attacker's odds are 68% and a weapon with a complexity of 15 is being used for the attack then the odds are increased to 75%.

Or: 15 divided by 2 = 7.5 (all decimal points get dropped so the result is then becomes 7), 68 + 7 = 75. If their roles are reversed here then the odds to hit are increased from 32% to 39%. 32 + 7 = 39.

If the attacking "monster" is the player then their character's armor expertise and weapon ability also get factored in. See the Eamon Deluxe 5.0 Players Manual for exact details on those factors.

Eamon Deluxe 5.0 limits the effect of all character/monster values during combat. So matter how high the values of agility, armor, weapon complexity, etc. are set, they can never have a larger effect than the highest "legal" limits.

The maximum limits of adjustments to combat odds are: Agility=30, Armor=7, Weapon Complexity=30, Weapon Ability (character only)=100.

For example a monster with an agility of 50/armor value of 10/weapon complexity of 100 will have the same odds adjustments as a monster with an agility of 30/armor value of 7/weapon complexity of 30. These limits quietly keep Eamon Deluxe grounded to all of Donald Brown's original Eamon formulas (and realistic) without placing limitations on character development and monster creation.

NOTE: When creating monsters, refer to APPENDIX 4 for data values.

5.2   ENTERING MONSTERS: NUMBER OF MEMBERS AND COURAGE

ALL monsters in Eamon Deluxe are group monsters, single monsters are just a 'group' of 1. This field is very useful. You can have "4 Rats", "12 Orcs", etc. without having to code more than one of them.

Courage is a monster's tendency to flee from a fight when injured and/or to pursue the player when they flee. Courage of less than 100 is the % chance that a monster will stay and fight. Courage of 200% will always pursue the player and fight to the death. If the monster is injured then courage is reduced by 5%, if the monster is nearly dead then courage is reduced by 10%. If a monster is in perfect health but the 'GROUP' number is less than the original size of the group (some of them died or fled) then the monster is considered injured.

NOTE: When creating monsters, refer to APPENDIX 4 for data values.

5.3   ENTERING MONSTERS: MONSTER ROOM NUMBERS

There are no special room codes for monsters. Either it's the number of the room in which it will be found or zero if you don't want it around until activated by special programming (such as a disguised monster).

5.4   ENTERING MONSTERS: ARMOR AND WEAPONS

Monster armor is the amount of hits absorbed or stopped per blow and is equivalent to the player armor system. Armor class is deducted from agility during battle (the weight of its armor slows it down).

Monster weapons are the number of the artifact that it is using for a weapon. A value of 0 means that it has natural weapons such as claws. A negative number means that it is unarmed.

Number of dice and number of dice sides is used to compute battle damage by the monster ONLY if it is using natural weapons. If it uses a weapon than the weapon will determine the battle damage. Example: If number of dice = 1 and number of sides = 8 then it will do 1D8 of damage, or from 1 to 8 points of damage. If the two numbers are 2 and 6 (2D6) it will do between 2 and 12.

5.5   ENTERING MONSTERS: FRIENDLINESS AND COMBAT CODE

Monsters in Eamon Deluxe by default have limited artificial intelligence. That is, they have little in the way of personality unless you program them to. The following fields give them what personality they do have.

Friendliness ratings are 1=Enemy, 2=Neutral, 3=Friend. Random chance of friendliness is assigned by using the percent chance + 100. Example: 140 indicates a 40% chance for friendliness.

Friendly monsters will smile at the player, follow them around, fight with them, and give them any possessions they carry. Neutral monsters will simply ignore the player.

The other attribute of the monster's personality is an Eamon Deluxe original field which can be a very powerful tool. The 'Combat Code' values are:

```
1.......The monster will be expressed as "ATTACKING" rather than using the
        random combat verbs.  (This should be used with monsters such as
        snakes, green slime, birds, etc. which attack in an unusual way)

0.......The monster will only fight with a weapon (or its natural weapons)
        This is the standard value for normal monsters.

-1......If there's no weapon around, the monster will use natural weapons

-2......The monster will never fight
```

5.6   ENTERING MONSTERS: USER FIELDS

USER fields are two reserved data fields included to make your own special programming a bit easier. They don't do anything at all unless you program routines which implement them.

6   ENTERING EFFECTS

'Effects' are stored in a random access file called EFFECT.DSC (LEN=255) and are there for you to use to store extra text, rather than cluttering up the program with it. To read and print your effects during the game use the code: "r = (effect no.) : GOSUB Effect". Where "r" is the number of the effect to be displayed. For examples of this, check out the TrollsFire routines in the Beginner's Cave. You can also substitute Effect1 and Effect2. Effect prints the Effect in whatever the current COLOR is. Effect1 prints it in COLOR 2 and returns with COLOR = 3. Effect2 prints it in COLOR 2 and returns with COLOR = 12.

NOTE: See section 2.2.2 on linking more than one effect together.

7   MODIFYING THE BASE DUNGEON PROGRAM

[Note: This section needs updating.]

While the size of the Eamon Deluxe 5.0 MAINPGM is slightly smaller than previous versions (and there is more memory available to the designer than before), it is much more condensed and actually includes a lot of enhancements and new features. INTRO is now a little larger in size than its previous versions because it now has more options pre-programmed into it for the designer to use (including the standard menu selection routines, and automatic VI Mode support).

Both the core module and SUB Init handle the loading and starting of the adventure. The startup routines have been streamlined to use as little code as possible and many common variables are expected to still be set to zero as it follows a tightly programmed order. Extreme caution should be used when modifying this section.

The framework of MAINPGM 5.0 requires the a%(0,x) subscripts to remain consistently set to zero and the size and order of the default command set to remain intact. See section 7.2 of the design manual for instructions on safely adding and removing commands.

All of the work above was to put your dungeon into a format that the Eamon Deluxe system can use. You can simply play the adventure straight after data entry is over. Select the test option and see how everything works out.

If you want special effects such as a magic sword (like TrollsFire in the Beginner's Cave) then you'll have to program them in. Start your adventure using the "Test" option from the design menu, then when the program starts up, hit control-break (control-Scroll Lock in DOSBox) and add your code. Because of the large amount of pre-programmed special artifacts, locations, etc., you can actually write a decent adventure without programming a single line of code.

However, a good dungeon needs a few special events; a GREAT dungeon is nothing but specials. The Eamon Deluxe Demo Adventure is an example of an adventure with no special programming but a large amount of special events.

NOTE: If you are adding new code, you should temporarily disable the ON ERROR code at the beginning of the MAINPGM. Simply change ON ERROR GOTO... to 'ON ERROR GOTO... When you are done testing make sure to enable it once again.

7.1 THE SYNONYM ROUTINE

This is a very simple, yet effective and powerful routine which helps you to hide things. For example, if a "secret door to the east" is embedded in a room, you don't want to give away the name to the player. This is where the synonym checker comes in. You can have your room description say something like "The north wall looks rather smooth compared to the rest of the room." Then, add this line to the "Synonym:" subroutine in MAINPGM.BAS:

```
IF RO = 12 then SY = 7: SY$ = "wall"
```

Where artifact 7 would be the secret door and room #12 would be the location. The secret passage in the Beginner's Cave makes a good example of using the synonym checker.

7.2 MODIFYING THE DEFAULT COMMAND SET

The framework of MAINPGM 5.0 has the default command set integrated on many levels which require that the current structure remains intact. However, it is possible to safely add and remove commands following the guides below.

A new command can be added by making four simple changes to the base program:
1. Locate the line in the core module that starts with "DATA 42,leather," and increase 42 to 43 to increase the total number of commands.
2. Add the name of your new command after "quit" on the DATA statement below.
3. Find the program label "Com3:" and, directly after "Quit" in the line below it, add the label that you want your new command to branch to. If you don't have a label yet, you can use ",Main" as a placeholder until you do.
4. Find the "Err0:" label and, in the line below it, change "FOR c = 33 TO nc" to "FOR c = 33 TO nc -1". Below that, change "(None)" to the name of your new command.

To safely remove one of the default commands, add a line to the very beginning of the SUB Main module with the code: c$(x)="" (where -x- is the number of the command to remove).

7.3 USING NEW VARIABLES

When adding your own special routines, you'll probably be needing to use special variables. Let me strongly suggest that you only use the d(x) and d%(x) variables and add no new ones to the MAINPGM. d(x) and d%(x) are sets of arrays each from 0-50 which were put in there for personal use. d%(x) is for INTEGER numbers (-32768 to 32767) and d(x) is for longer, floating point numbers.

The reason why you should use these instead of naming new variables is that the SAVE and RESTORE commands work by making a "snapshot" of the adventure variables in a data file. The d(x)/d%(x) arrays are automatically saved by default, but any new variables will have to be added to these routines and could cause trouble in the long run. It has also become standard practice to keep track of what each array means by adding comments to the beginning of MAINPGM. That way you'll never forget, and others can better understand your programs as well.

NOTE: Use LINE INPUT rather than INPUT when restoring strings from files, in case they have any commas or other 'illegal' characters in them.

NOTE: The d%() arrays are INTEGER variables and can't hold a number higher than 32767 or less than -32768.

7.4 BASE PROGRAM SUBROUTINES

You can save yourself a lot of time by utilizing the subroutines already in the MAINPGM. You're likely to find most everything you'll need. See Appendix 2 for a complete list of subroutines.

7.5 TELEPORTING MONSTERS IN/OUT OR CHANGING FRIENDLINESS

Whenever a monster comes, goes, or suddenly becomes a friend/enemy, you must make sure the NBTL() variables are set or things could get buggy. To avoid problems, simply GOSUB EnemyCheck after all such events.
Example: m%(12,11) = 3: GOSUB EnemyCheck

7.6 DAMAGING A PLAYER/MONSTER

It is not uncommon to have events that cause harm to either the player, a monster, or both. Set the following variables: a (1=armor can absorb some of the damage, 0=armor ignored); d2 (amount of damage); df (monster number of victim, 0=player) and GOSUB MStatus.

Example: "df = 0: d2 = 5: a = 0: GOSUB Mstatus" will do five points of damage to the player, ignoring his/her armor.

7.7 SPECIAL EVENT DELAYS

Because different computers run at different speeds, all pauses and timed effects require different delay values to work properly. Eamon Deluxe provides its own variables to make it simple for you. The SETTINGS.DAT file is written to all adventures when being played or tested. This small file holds 13 variables: DLY#, CC1#, CC2#, CC3# and 10 numbers that describe the current Eamon Deluxe System Settings.'DLY#' is roughly the number of times to repeat a FOR/NEXT loop to delay for about one second.

```
Examples:  FOR second# = 1 to dly# : NEXT
           FOR halfsecond# = 1 to dly# / 2 : NEXT
```

Note: Always use double-precision variables in your delay loops!
(e.g. "FOR X# = 1 TO..." instead of "FOR X = 1 TO...").

CC1#-CC3# are delay values for the standard twirly-cursor input routines. The values are set to your computer's speed when you first install Eamon Deluxe and can be reset via the Main Menu's Control Panel option. You MUST base any delays in your programs on this variable or they will be too fast/slow on different computers. For an example, check out the USE SHOVEL code in The Lair of the Minotaur (adventure #2, The Don Brown Adventures) or the Delay SUBS in the EDXDEMO.BAS file (found in either EAMONDX\MAIN or EDX\C\EAMONDX\MAIN).

EAMON DELUXE 5.0 NOTE: When running in V.I. Mode delays should be ignored and skipped over because Eamon Deluxe will be running in a terminal rather than DOSBox and delays will never work properly under Windows anyway and can even cause your programs to hang or crash.

9. USING THE INTRO PROGRAM

[Note: This section needs updating.]

"INTRO.BAS" is the first program which is run by the Main Hall/Test Program. All you have to do is add your name, adventure name and intro story after the "Intro:" label. INTRO automatically puts the name of the character in the "name$" variable, the character's sex (0=male,1=female) into the SEX variable, and a quotation mark in the q$ variable for you to use. You can refer to the intro to "Lair of the Minotaur" on The Donald Brown Adventures which makes good use of the SEX variable.

You need to change the variables "adventure$" and "author$" at the beginning to your name and the adventure name. Additionally, INTRO has some useful subroutines for you to use. "Title:" clears the screen and displays the adventure and author names, "KeyWait:" prompts the user for a keypress then goes to title, "Center:" centers and prints the variable "a$" on the 80-column screen.

The EamonDXMenuM SUB is now included in the base INTRO program to use when you want to offer the player choices to pick from. This routine is already set to be compatible with V.I. Mode as well. See the comments in the INTRO program for info on how to use it.

10. HAVING MULTIPLE ADVENTURES IN ONE DATABASE

[Note: This section needs updating.]

14. KNOWN FLAWS IN THE EAMON DELUXE SYSTEM

NEVER USE SUB DIRECTORIES WITHIN YOUR ADVENTURES! The Eamon Deluxe 5.0 system use a pre-defined directory tree and errors ranging from minor to serious will occur if that tree is modified or new adventure directories are added which contain sub folders.

15. FINAL NOTES ON DESIGNS

[Note: This section needs updating.]

You can learn a lot by tearing apart other people's adventures and seeing how they did things. More often than not you can save yourself a lot of time writing and debugging by using a routine from a different MAINPGM as an example. I myself can't think of a greater honor than somebody thinking a routine of mine good enough to steal! And with the adventure count for Eamon Deluxe approaching 300, it is safe to assume that someone has already had the same ideas as you and written working code for it.

Try to have some point to your adventure. Free a princess, steal a magic sword, slay an arch-fiend, anything. Themes like "You saw a cave and decided to explore it" get boring fast.

Many people have griped over the years about technological barriers being completely ignored by designers. In more simple terms, they hate "walking down the corridor of a medieval castle in full plate armor and being suddenly mowed down by a Nazi with an Uzi." I'm not bothered by this myself, Eamon supposedly being a strange magical world where anything goes, but a lot of people feel otherwise. Just a point to consider.

Don't feel at all limited to what's been done already. The beauty of a computer-based role playing system is that is completely open-ended. If you work hard enough, most anything can be done. Monsters that can hold complete and intelligent conversations? Not impossible, you would just need a good parser and a large database.

Try and use known characters in the right context. If Hokas Tokas from the Main Hall is in your adventure, he should act like he normally does. Feel free to expand his character of course, just be reasonable. The player will see these guys as soon as they return to the Main Hall so they can't become arch-villains or anything. The player should also not be allowed to attack them and they should never "die" (have them "vanish in a puff of smoke!" instead). If you use characters from other adventures try and make them act and look like the original author did.

People really hate traps of the "Zap! You're dead!" type. Try to avoid loading your adventures with instant death traps. A better idea is a trap which simply does damage to the player or gives them a nice startling brush with death. Killing the player off without warning generally takes some of their interest out of the game and is a form of sloppy programming anyway.

Note that instant death traps were a large part of early Eamons (and most commercial text adventures) and they CAN be included in the game, by the hundreds if you wish, as it is after all YOUR adventure. Which brings us to the final subject:

15.2 SUBMITTING AN ADVENTURE AND GETTING AN OFFICIAL NUMBER

[Note: This section needs updating.]

Last but not least, don't be afraid to break any of the above suggestions. If you truly believe that your dungeon will be better, do anything you please; it's a world of your own creation. The worst that can happen is that other people may give it bad reviews or not want to play it. The only thing that truly matters in YOUR dungeon is that it is just the way you want it.

Please submit new adventures to eamondeluxe@gmail.com so your creation can be added to the Official Adventure list. Once an adventure is submitted it will be played through and checked for critical bugs and you will get a response with an Official EDX number and anything that the play tester(s) found noteworthy. It is the completely up to the author if they want to make any suggested edits or release it as it.

16. CREDITS AND LICENSE INFO

[Note: This section needs updating.]

Credit goes to the developers of the original Apple II Eamon system, Donald Brown, John Nelson, Rick Volberding and Thomas Zuchowski. While obviously and intentionally based upon Classic Eamon, Eamon Deluxe is a unique system written and developed entirely by Frank Black Productions. It is intended as a tribute to Classic Eamon, a preservation effort, and a gateway for the continued creation and enjoyment of Eamon adventures in a modern environment.

Along with those mentioned above, special thanks also go out to:

Matthew Clark for creating and maintaining the official home of the Eamon Adventurer's Guild (www.EamonAG.org). Matthew and his website have long been an indispensable resource to the Eamon community.

The DOSBox Team (www.dosbox.com) for providing me with one of the tools needed to continue the development and cross-platform distribution of the strange, continually-mutating creature that is Eamon Deluxe (still in its native environment, long after support for that environment was abandoned by its own creator). DOSBox 0.74 is free software distributed under the GNU General Public License.

Luke Hewitt and Patrick Moen for their extensive help in the inspiration, development and testing of VI Mode.

John MacArthur, Jared Davis, Thomas Ferguson and Derek C. Jeter who were the first authors to pen adventures using the Eamon Deluxe system. Thomas deserves extra thanks for all of his work with recovering lost treasures, converting classic adventures, and a multitude of other contributions to the Eamon community.

And, of course, all of the game authors out there who contributed their own entries into the massive and varied Eamon adventure library. Whether it be a 26-room, 5-monster cave or a Middle-Earth epic, every Eamon adventure is a fun and fantastic journey into the imagination of its creator and is a piece of gaming history.

Donations are very much appreciated but never required; all I really ask is that you have fun.

APPENDIX 1: MAINPGM 5.0 - COMPLETE VARIABLE LIST

This appendix is provided as a reference for game authors who wish to program special events into a new adventure using the Eamon Deluxe 5.0 system and includes a complete list of every variable used by the MAINPGM. All of the variables are of the "SHARED" type.

Note: Many variables used in previous versions of the Eamon Deluxe MAINPGM have been renamed or eliminated with the 5.0 revision. For information about older versions, please refer to the Eamon Deluxe 4.7 Reference Guide.

```
   a$(na): Artifact names.
 a%(na,8): Artifact index number, artifact data fields:
              0 = Seen flag (0=No, 1=Yes)
              1 = Value
              2 = Type
              3 = Weight
              4 = Room
            5-8 = (See section 4.4 on Artifact Types)
 bv$(5,2): Random combat verbs. (Weapon type, verbs)
   c$(nc): Command set.
   d%(50): Reserved for game designer to use with special events programming.
    d(50): Reserved for game designer to use with special events programming.
   gm$(5): Current names used to describe saved game positions.
   h$(nh): Titles of available hints.
 h%(nh,2): Hint text pointers (1=starting record, 2=total number of records)
   m$(nm): Monster names.
m%(nm,15): Monster index number, monster data fields:
              0 = Seen Flag (0=No, 1=Yes)
              1 = Hardiness
              2 = Agility
              3 = Number of members in group (1=Individual monster)
              4 = Courage (%)
              5 = Current location
```

```
               6 = Combat Code (See section 5.5)
               7 = Armor (hits absorbed)
               8 = Weapon artifact pointer (0=Natural weapons)
               9 = Number of dice for natural weapons
              10 = Number of dice sides for natural weapons
              11 = Personality Type (1=Enemy, 2=Neutral, 3=Friend)
              12 = Original size of group
              13 = Current amount of damage taken in battle
              14 = Reserved for use by designer. (USER #14)
              15 = Reserved for use by designer. (USER #15)
   NBTL(4): Hardiness totals (by Personality Type) of all monsters in room.
    r$(nr): Room names.
 r%(nr,11): Room data. (0=Seen flag, 1-10=Connections, 11=natural light level)
  rb$(5,1): Random combat fumble verbs. (Target's weapon type, verbs)
    s%(10): Current Eamon Deluxe 5.0 system environment settings.
    sa%(8): 1-4=Current spell abilities, 5-8=Maximum spell abilities (%).
              (1/5=Blast, 2/6=Heal, 3/7=Speed, 4/8=Power)
   sex$(1): Gender strings. (0="Male", 1="Female")
    sm$(3): Smile response verbs. (1-3 by Personality Type)
    wa%(5): Character's current weapon abilities (%).
    wp$(5): Weapon type name strings. (Axe/Missile Weapon/Club/Spear/Sword)

   A0$: Subject of the player's command exactly as entered (before LCASE$ and
        parsing) Used for the SAY command and Quick Saved game descriptions.
    a$: Generic string used almost everywhere.
     a: Common generic variable, most often used as an artifact pointer.
    a2: Generic variable, used when common generics are already in use.
   ac%: Character's Armor Class at startup. Updates to Armor Class of whatever
        armor artifacts the character is wearing when exiting the adventure.
  adj1: "Adjustments" or database start and end pointers for multi-adventure.
        databases. adj1=Rooms, adj2=Artifact, adj3=Monsters, adj4=Effects
   adv: Number of adventure being played in a multi-adventure database.
   ae%: Character's Armor Expertise.
   af%: Armor Factor: Effect of character's current armor vs. Armor Expertise
        upon combat odds (offensive only).
   ar%: Artifact pointer for armor being worn by character. (0=None)
  bank: Amount of gold character has in the bank.
     c: Number of last command entered by player.
  CC1#: Used along with CC2# and CC3# for twirly cursor delays. Currently
        loaded from SETTINGS.DAT but not used by MAINPGM 5.0.
   ch%: Character's Charisma.
   cz$: Last command entered. Used when Enter is pressed to repeat commands.
     d: Generic variable.
    d2: Generic variable.
    db: Pointer to artifact number of first dead body when bodies are used.
    df: Monster pointer of defender in battle (0=character).
   die: Character's Life or Death flag (0=alive, 1=dead)
  dly#: Delay value for a one second pause under current System Settings.
   em%: Generic variable, mostly used by the search routines ("Embedded").
     f: Common Generic variable, commonly used as "found" flag by the search
        routines. (0=Not Found, 1=Found, 2+=Multiple matches found)
  gold: Amount of gold coins character is carrying.
   ha%: Generic variable, most commonly used by the search routines ("Have").
    k$: Generic string, currently only used by the error recovery routine.
     k: Generic variable, used when common generics are already in use.
     l: Line Counter. Number of lines printed since last screen pause.
    l2: Line Counter 2, used by the Main Loop for text formatting. (0=Last line
        was text, 1=Line was blank, 3=Force break after artifact/monster name)
   ls%: Artifact pointer of current light source.
    lt: Light. Visibility flag of current room. (0=Dark, 1=Light)
     m: Common generic variable, most often used as a monster pointer.
   mc%: Active Member Counter for group monsters in Pfoe and Battle routines.
    na: Total number of artifacts including Character's starting weapons/armor.
   naf: Total number of artifacts before Character's starting weapons/armor.
    nc: Total number of valid commands.
    nd: Number of directions character can move in. (6=Standard, 10=Extended)
    ne: Total number of Effects in EFFECT.DSC file.
    nh: Total number of hints available in the hint files.
    nm: Total number of monsters.
    nr: Total number of rooms.
 odds%: Generic, most commonly used as hit/miss odds during combat.
    of: Generic, most commonly used as Attacker Pointer during combat.
    q$: Holds a quotation mark (CHR$(34)) for easily printing text in quotes.
     r: Common generic, most often used as record pointer for data files.
    r2: Room character is moving into.
    r3: Room character last exited.
    rl: Random number returned from RollDice routines.
    ro: Room character is currently in.
     s: Generic variable, often used when common generics are already in use.
    s$: Subject (or object) of player's command input.
  s2$: Second Subject (or second object) of player's command input.
 sex%: Character's sex (0=Male, 1=Female)
  sh%: Artifact pointer for shield being worn (used) by character. (0=None)
   sl: Generic, commonly set as String Length in search and synonym routines.
Speed: Speed Spell Flag/Counter for number of turns left before it expires.
  sy$: Generic string, commonly used as Synonym routine search strings.
   sy: Generic, commonly used as artifact pointer for Synonym routine.
   v$: Verb (or actual command) of player's command input.
    w: Generic, used as artifact pointer for Offender's weapon during combat.
  wh%: Generic variable, most commonly used by the search routines ("Where").
   wt: Sum of the weight of all artifacts character is carrying and wearing.
   x$: Generic string, commonly used for player input outside of the Main Loop.
    x: Common generic variable.
   x1: Generic variable, used when common generics are already in use.
   x2: Generic variable, used when common generics are already in use.
```

APPENDIX 2: MAINPGM 5.0 - PROGRAM LABELS (ROUTINES AND SUBROUTINES)

This appendix is provided as a reference for game authors who wish to program special events into new adventures created using the Eamon Deluxe 5.0 system. It includes a complete list of program labels and describes all of the MAINPGM 5.0 routines and subroutines.

Note: Almost every line in the MAINPGM was rewritten for Eamon Deluxe 5.0. While the interface still appears very similar from a player's perspective, "under the hood" it is extremely different. Many program labels found in previous versions of the MAINPGM have been moved, renamed or deleted. For information about the program labels in the older versions, please refer to the Eamon Deluxe 4.7 Reference Guide.

MAINPGM 5.0 SUBROUTINES (access with GOSUB except when noted)

Common text printing utilities

```
Effect: Prints the text from record -r- of the EFFECTS.DSC file.
Effect1: Prints effect in color 2 and then resets text color to 3.
Effect2: Prints effect in color 2 and then resets text color to 12.

Lp0: Prints -a$- with proper line breaks for the 80-column text screen.
Lp0Err: Error message for strings not printable by Lp0.

Lp8A: Prints -a$- in color 2 with line break and branches to Main. (Use GOTO)
Lp8: Prints -a$- with line break and branches to Main. (Use GOTO)
Lp9A: Prints -a$- in color 2 with line break and branches to Pfoe.(Use GOTO)
Lp9: Prints -a$- with line break and branches to Pfoe. (Use GOTO)

Lp10: Prints -a$- with line break afterwards.
Lp10A: Prints -a$- in color 2 with line break afterwards. Resets color to 3.
Lp10B: Prints -a$- with line break afterwards. Resets color to 3.
Lp10C: Prints -a$- with line break afterwards. Resets color to 12.

Lp12: Prints line break and branches to Pfoe. (use GOTO)
```

Player input utilities

```
Af0: Verifies that the player wants to act hostile towards a non-enemy.
AF1: Prompts for a yes or no input. Resets -s$- to "y" or "n".
Af2: Verifies that the player wants to extinguish a lit artifact.

CheckSub: Prompts for subject of player command if none was entered.
CheckSub2: Prompts for second subject of player command if none was entered.

Synonym: Attempt to match player input to user-added artifact name synonyms.
Synonym1: Checks for match between -s$- and -sy$-, resetting -s$- to the
name of artifact -sy- if a match is found.
```

Line counter/scroll pause utilities

```
Lp1: Does a PRINT and increments line counter by 1.
Lp2: Increments line counter by 1.
Lp3: Does a PRINT and increments line counter by 3.
Lp4: Does a PRINT and increments line counter by 2.
Lp5: Checks line counter to see if screen is full and pause is needed.

Lp6: Pauses text printed to screen until a key is pressed.
Lp6a: Resets line counter to 0. Clears screen if VI Mode is enabled.

Lp7: Clears screen and resets line counter to 0.
```

```
 Grammar utilities
-------------------
Caps: Sets -a$- as -a$(a)- and falls into Caps1
Caps1: Capitalizes -a$- and adds a space to the end of it.

Pu2: Sets -a$- as the name of monster -m-.
Pu2a: Sets -a$- as the name of monster -x-.
Pu3: Makes -a$- possessive with a space at the end. (e.g. "Bob's ", "Brutis' ")


 Common action utilities
-------------------------
RollDice: Resets -rl- with a random number between 1 and 100.
RollDice2:  Resets -rl- to a random number between 1 and current value of -rl-.

ItemTake: Check if player is carrying an item before manipulating it.
If not carried, attempts to GET it with a branch to ItemGet3.

WTRem: Decreases carried weight counter if artifact -a- is being carried.

ContWT: Calculates the weight any items inside of artifact -a- and returns with
the total value in -w-.

EnemyCheck: Resets -NBTL(1-3)- to total hardiness of player and all monsters in
the current room based upon friendliness value. Also Determines friendliness
values for monsters if not already set.

SetFactors: Resets -af%- a player's armor factor and -f- to charisma factor.

MoveMons: Moves appropriate monsters from current room to the room that the
player is moving into. Also Determines friendliness values for monsters if not
already set.

CheckNX: Checks current room connections and calculates number of exits.
Resets -k- to total exits for monsters (valid room numbers only) and -w- to
total exits for player (valid room numbers plus "special" negative moves).

CheckDoor: Resets -d- to the room number that a door connects to if the door
is open and not hidden, otherwise sets -d- to zero.

Ups: Prints message when various player abilities increase.

LightOut: Resets light level variables to default of room and displays message.

SpellCast: Determines if attempted spell -s- was successful and adjusts spell
ability variables accordingly. Resets odds% as success flag (0=no, 1=yes).

GameDIR: Loads info about saved games. Returns with number of saved games in
the -a- variable and updates the gm$() array with saved game names.


 Search routines
-----------------
Asearch: Searches artifacts carried by player or located/embedded in room.
Asearch1: Searches artifacts located or embedded in room only.
Asearch2: Searches artifacts carried by player only.
Asearch3: Searches artifacts worn by monster -m-. (0=Player)
Asearch4: Searches for artifacts with locations matching -ha%- only.

Asearch10: Executes Synonym subroutine then compares player input with the
names of all artifacts in locations specified by -wh%-, -em%- and -ha%-.
Routine exits with -a- and -f- set depending upon results of search. If no
match was found -a- and -f- are both set to zero. If more than one match was
found then -a- is set to zero, -f- is set to number of matches and an error
message is printed. Upon a single match, -a- is set to the artifact number and
-f- is set to 1; if artifact matched is embedded in current room (and current
player command is not CLOSE or EXAMINE) then the artifact is placed in the
room and it's description is printed.

Msearch: Sets -wh%- and -ha%- to current room and branches to Msearch0.
Msearch0: Compares the names of all monsters in locations specified by -wh%-
and -ha%- with player input. If a match is found, -m- is set to monster number
and -f- is set to 1, otherwise both are set to zero.
Msearch1: Direct jump point for NEXT statement in monster search loop.


 Common error messages
-----------------------
```

```
Err0: Displays valid command list.
Err1: "You don't have it and it's not here."
Err1A: "You don't have it" plus whatever -a$- is set to.
Err2: "You must first open it."
Err3A: "You can't [command] that."
Err3: a$ = "You can't [command]" plus whatever s$ is set to.
Err4: "Nobody here by that name!"
Err5: "You don't need to." (Resets -f- to zero)
Err6: "You're wearing it. REMOVE it first." (Resets -f- to zero)
Err7: "I don't follow you." (Resets -f- to zero)
Err9: Prints error message stored in -a$- (only if -f- is less than 2) and
then branches to Main.


 Utilities for reading/printing text from description files
------------------------------------------------------------
DiskRead1: Prints the text from record -r- of the ROOMS.DSC file.
DiskRead2: Prints the text from record -r- of the ARTIFACT.DSC file.
DiskRead3: Prints the text from record -r- of the MONSTERS.DSC file.
DiskRead4: Checks end of -a$- for text-chaining wildcards. Sends -a$- to Lp0.
If a wildcard was found then resets -r- and branches to Effect.
DE: Error message for invalid file records.


 Combat and monster health related subroutines
-----------------------------------------------
Mstatus: Adds -d2- to damage taken value of monster specified by -df- then
prints current health status. Exits if current damage taken doesn't exceed
monster's Hardiness, otherwise branches to Mdead.

Battle: Start of the combat subroutine. Monster -of- attempts to attack monster
-df-. Checks Combat Code and Weapon status of -of-, exiting if they are found
unable to attack. Calculates various factors for the attacker and their target
and compares resulting odds with a 1-100 dice roll. Prints attack message with
random combat verbs and branches to Battle1 if the attack was successful,
otherwise executes the routines for missed and fumbled attacks.

Battle1: Checks for skill increase if -of- is the player.
Battle2: Determines hit potency depending upon a roll from 1-100.

Damage0: Calculates amount of damage hit will cause using preset weapon
dice (-d-) and sides (-s-). Subtracts value of target's armor if preset armor
flag (-a-) is set to 1. Sets -d2- as damage value.
Damage1: Prints appropriate message if -d2- is less than 1 and exits,
otherwise falls into Mstatus.

Mdead: Prints message and starts monster death routines. If -df- is a member
of a group then group size is reduced, weapon is dropped (if applicable)
and a branch is done to Mdead2. If -df- is a "group" of 1 then weapon value,
room number, damage taken and group size (if applicable) are reset and any
items worn or carried are dropped. If the dead body pointer (-db-) has been
set (and it points to a valid artifact) then body artifact is placed in room.
Mdead2: Adjusts NBTL flag related to monster's friendliness and exits routine.

GetWep: Routines for unarmed monsters to search for weapons. First looks for
recently dropped weapon, then carried weapon, then weapons in room. Always
selects the best weapon available.
GetWep1: Exit point when monster picks up a weapon.


 Other subroutines
-------------------
The following subroutines are helpers to standard routines and are described
in the MAINPGM 5.0 Routines list below (but are still accessed with GOSUB):
Drop2, Flee1, ItemGet3, Inv3, Inv4, Smile1, SpellCast1, Wear1, Wear2


MAINPGM 5.0 ROUTINES (access with GOTO)

CORE MODULE AND SUB INIT PROGRAM LABELS:

ErrHandler: Error handling routine.
Pwr: Data pointer for the random text strings used by the POWER routine.

CheckDup: Checks artifact names and adds "#" to the end of any duplicates so
that the command parser can tell the difference between them.


SUB MAIN PROGRAM LABELS:
```

Main: The start of the Main Loop, all commands and events return here. Decrements counter for light source and/or SPEED spell if active and partially restores spell abilities if needed.

Main1: Prints "too dark to see" message and skips to command input routine (Com1) if room light value is 0, otherwise sets text color to 12 (Main Loop standard) and prints room info.

Main2: Prints monsters in room info.
Main2A: Skip directly to NEXT monster in info loop.
Main3: Prints artifacts in room info.
Main3A: Skip directly to NEXT artifact in info loop.

Com0: Reserved for special, user-programmed events.

Com1: Command prompt. Gets input from player and attempts to parse command, subject and second subject.

Com2: Attempts to match player input with valid command set. Branches to Err0 if no match is found.

Com3: Branches back to Main for most commands if light level is zero, checks spell success and branches to Pfoe upon failure if command was a spell, otherwise branches to appropriate routine for command entered.

Pfoe: "Pick Foe". Start of the "Every-Round Events Loop". Sets text color value to 3 (standard events color). Branches directly to Pfoe3 if no enemies are in the room, otherwise executes Combat Loop.

Pfoe0: Start of the Combat Loop. Checks all monsters in database for location and friendliness, skipping to Pfoe2A if monster is not in current room or has a neutral status. Executes fleeing routine if monster fails courage check, otherwise branches to SkipFlee.

SkipFlee: Configures combat data for current monster in Combat Loop.

Pfoe1: Increments group counter if needed and selects an appropriate foe.

Pfoe2: Executes the Battle subroutine then returns to Pfoe1 unless no more enemy monsters are left in the room.

Pfoe2A: Branches to Pfoe0 until Combat Loop counter reaches last monster.

Pfoe3: Reserved for special user-programmed events, branches back to the Main Loop (Main) when finished.

Move: Start of directional commands routine.

MoveCheck: Processes connection value of direction player is trying to move in. Skips to Move2 if connection is a valid room number, otherwise checks if value is an unhidden door or pre-programmed special connection. If no match is found then it branches back to Main with an error message.

Move2: Resets appropriate variables to move player and any monsters who follow into a new room and then branches to Main.

Flee: Flee routine for player. Executes Flee1 subroutine if no direction is specified then branches to MoveCheck.

Flee1: Randomly checks the connection values of the current room, selecting the first one found which leads to a valid room number. Returns with room value in r2 variable. Note: Uses the variable -m- to determine appropriate directions (only the player can FLEE to pre-programmed special connections).

ItemClose: Routine for the CLOSE command.

Drink: Routine for the DRINK and EAT commands.

Drop: Routine for the DROP and DROP ALL commands.
Drop2: Adjusts any variables related to an artifact (-a-) being dropped.

Examine: Examine routines for rooms and artifacts.
Examine1: Examine routine for monsters.

ItemGet: Routine for the GET and GET ALL commands.
ItemGet3: Adjusts any variables and/or performs special events related to the artifact (-a-) the player is attempting to get.

Light: Routine for the LIGHT command. Turns appropriate artifacts on and off.

ItemOpen: Routine for the OPEN command.

ItemPut: Routine for the PUT command.

ItemRead: Routine for the READ command.

Ready: Routine for the READY command.
Ready1: Prints message that weapon has been readied and updates related variables, branching to Pfoe when finished.

Remove: Routine for the REMOVE command.

Use: Routine for the USE command.

Wear: Routine for the WEAR command.
Wear1: Prints item worn message, updates artifact room location to -999 and branches to Pfoe.
Wear2: Error messages relating to the WEAR command.

Attack: Entry routine for the ATTACK command. Checks for readied weapon and tries to match input subject with monsters in room. Branches to AttackItem if no match is found.
AttackMon: Player attacks a monster routine.
AttackItem: Player attacks an artifact routine.

Free: Routine for the FREE command.

Give: Routines for the GIVE command. Checks subjects of player input and branches to Give1 unless player is attempting to give money.
Give1: Routine for the GIVE command related to artifacts.
Give2: Updates enemies in room flag and branches to Pfoe.
GiveDrink: Routine for the GIVE command related to food and beverages.

Request: Routine for the REQUEST command.

Smile: Routine for the SMILE command.
Smile1: Prints a message based upon the friendliness value of monster -m-.

Inv: Routines for the INVENTORY command. Checks for subject of inventory and branches accordingly.
Inv1: Special items worn list used only for player.
Inv2: Lists artifacts which are in the possession of monster -m- along with certain select details. Reports current health status and branches to Pfoe.

Inv3: Alt. entry for Inv4. Displays message and returns if item is closed.
Inv4: Lists the contents of a container artifact specified by -a-.

Status: Routine for the STATUS command. Calls the StatDisplay module and branches back to Main when done.

Look: Routine for the LOOK command.

Say: Routine for the SAY command.

Blast: Routine for the BLAST command.
SpellCast1: Prints "ZAP! Direct hit!" message for blast spell.
Heal: Routine for the HEAL command.
Speed: Routine for the SPEED command.
Power: Routine for the POWER command.

Hints: Routine for the HINTS command.

SGSave: Routine for the SAVE command.
SGSave0: Checks user input to determine slot number and name to save game data under. Cancelation requests branch to Main and invalid slot numbers branch to SGSave.
SGSave0A: Routine which saves current game data.

SGRestore: Routine for the RESTORE command.

Quit: Routines for the QUIT and QUIT HALL commands.

EndGame: Start of exit routines. Verifies whether player wishes to leave the adventure or not.

EndGame0: Alternate entry for exit routines used to skip player verification. Adjusts select variables and checks amount of weapons the player is leaving with, branching to EndGame1 if the amount is less than 5.

SellWep: Lists carried weapons and prompts for one to sell. Repeats if needed.

EndGame1: Displays buyer info and value of all artifacts in the players possession except kept weapons. Exits the SUB MAIN module when finished.

Dead: Displays options available if a character has died. Either restarts the adventure from the beginning, exits the SUB MAIN module with a dead character or branches to SGRestore.

APPENDIX 4: AVERAGE DATA VALUES FOR MONSTERS AND ARTIFACTS

A standard range of values was developed while converting the Apple II Eamon library to Eamon Deluxe. In the days of Classic Eamon, when anarchy reigned throughout the realm of game design, the same sword could be worth 5 gold pieces in one adventure and 500 in the next! The current Eamon Deluxe standards system was developed by referencing the Classic Eamon game manuals, examining the original set of adventures and Main Hall written by Donald Brown himself, and then averaging the results.

ARTIFACT VALUE, TYPE AND WEIGHT

Artifacts with a special weight value of either -999 or 999 can never be picked up or moved under any normal circumstances. The difference between the two values is that -999 gives the message "You can't get that", while 999 gives the message "Don't be absurd!"

Junk weapons: Broken swords, bent spears, dull knives, etc..
Average weapons: "Good" items that can be bought at the Main Hall Weapons Shop.
Exceptional weapons: Crossbows, well-crafted swords, exotic or magic artifacts.

An extra 10-20 gold pieces should be added to the value of weapons with a Complexity greater than 20% and/or a total damage potential of 2D8 or higher. For special items (like laser cannons, etc.) use your own judgment.

	Value	Type	Weight
Junk weapons	0-10	2	5-20
Average weapons	15-25	2	15-25
Exceptional weapons	30-35	2/3	(Compare to items of similar size)
Gold bar	50	0	50
Silver bar	25	0	25
Bag of 100 coins	100	0	30-60
Small gems	50-75	1	1-3
Giant emerald	75-120	1	5-10
Chairs, small tables	0-20	1	30-60
Beds, couches	0	1	999
Stone statue	0	1	-999
Small chest	0-10	4	5-10
Large chest	0-20	4	30-60
Unmovable chest	0	4	999
Torch (50+ turns)	3-5	5	5
Lantern (100+ turns)	5-12	5	5
Healing potion (+5)	35	6	6
Healing potion (+15)	50	6	6
Food items (perishable)	0	9	(Compare to items of similar size)
Wine/liquor/ale	5-50+	6	(Varied)
Messages, notes	0	7	0-1
Books (average size)	0-4	7	10-15
Books (heavy)	0-4	7	15-25
Books (rare or magic)	5-45+	7	10-15+
Doors, gates, etc.	0	8	-999
Bound/disguised monsters	0	10/12	-999
Leather armor	75-100	11	15
Chain armor	120-200	11	25
Plate armor	300-350	11	35
Standard shield	50	11	15
Overcoats	1-8	11	3-8
Hats, gloves, shoes, etc.	1-8	11	1-3
Watches, rings, necklaces	(Varied)	11	1-5+
Dead bodies	0	13	(Varied)
Dead rare animals	5-20+	13	(Varied)
Bodies (wanted criminals)	25-100+	13	(Varied)

AVERAGE MONSTER RATINGS

Weak monsters: Rats, wimps, kobolds...
Medium monsters: Petty thugs, orcs, goblins...
Tough monsters: Giants, trolls, skilled warriors...
Exceptional monsters: Dragons, demons, special characters...

	Weak	Medium	Tough	Exceptional
Hardiness:	2-8	9-15	16-30	31-60
Agility:	5-9	10-16	17-24	25-30
Courage:	80-90%	95-100%	200%	200%
Armor:	0	1	2-3	3-7+
Damage:	1D2-1D5	1D6-1D10	1D10-2D6	2D8+

On the chart above, Damage means either natural weapons or the dice and sides of the monster's initial weapon. If you want a monster to do a specific number of damage points, set the Damage values to Dice=Amount of damage and Sides=1. 10D1 will always roll out to 10 Damage Points.

If you want your monster to use both natural weapons AND artifact weapons then set their Combat Code to -1.

If you want your monster to be passive (they can be attacked, but won't fight back under any circumstances), set their Combat Code to -2.

If you have a group of monsters, such as "6 Orcs", you may want to scale them down to be weaker than usual since up to 5 members of a group can attack during every round of combat.

APPENDIX 5: MISCELLANEOUS REFERENCE INFORMATION

ARTIFACTS: DEFAULT SPECIAL ROOM LOCATIONS

-1	Carried by player
-(Monster# - 1)	Carried by Monster #
-999	Worn by player
-(Monster# -1000)	Worn by Monster #
Container Artifact # + 1000 ..	Inside artifact #
Room # + 2000	Embedded in a room #

WEARABLE ARTIFACTS: ARMOR AND CLOTHING TYPE VALUES

Armor Class	Clothing (Armor Class 0) Types
0 Clothing	0 Armor, shields, average clothes
1 Shield	1 Overclothes (coats, capes, etc.)
2 Leather armor	2 Shoes, boots
4 Chain Mail armor	3 Gloves
6 Plate Mail armor	4 Hats, headwear
8 Magic armor	5 Jewelry
10 Magic armor	6 Undergarments

COLOR NUMBERS AND DEFAULT USES

1 Blue	6 Brown	11 Light Cyan
2 Green (special events)	7 White	12 Light Red (Main Loop)
3 Cyan (standard events)	8 Gray	13 Light Purple
4 Red	9 Light Blue	14 Yellow (extra special events)
5 Purple	10 Light Green	15 Bright White (extra special events)