AsciiQuest Editor v1.0

(Version 1.0 documentation)
Programmed and maintained by Jace Masula 2005 / QBASIC LAB
Special thanks to : Ryan, Jocke for their help and inspiration.
visit my site at http://QB45.think-new.com

ASCIIQUEST is an ascii game creation package. It contains everything you need to created and play you very own ascii character rpg / puzzle games. You can create maps that are scripted to events. The easy to use event scripting allows you to build simple or complex game events. There are a few features and a few limitations of **ASCIIQUEST**. Before you get going on your first game, make sure you will be able to accomplish what you want using the **ASCIIQUEST** editor. The way that the engine has been designed should allow you to create some pretty cool ascii rpg / puzzle games. One great thing about the engine, is any character saved from a previous game, can be loaded into a newer version of the game and retain all items, weapons, armor, magic , etc…. This allows you to create smaller games that have multiple versions and or chapters.

ASCIIQUEST is easy to use and the game engine makes the games you create easy to play. This program is 100% free and only displays 2 small lines of credits at the bottom of the game engine credits screen. The editor is mouse driven and the game engine is keyboard only.

The AsciiQuest Game editor / engine
These 2 main components will allow you to create and play your very own ASCII character RPG type puzzle games. This simple manual will help you get started creating your first game. Keep in mind that the editor and engine are somewhat simple… if you are looking for an advanced RPG game maker, this is NOT it. However, the **ASCIIQUEST** game engine can do some neat stuff. (once you get the hang of how it works)

Frequently Asked Questions about ASCIIQUEST v1.0

- **Can you animate stuff in the game?**
  Yes, with some free time and plenty of patience, you can animate just about everything in the game. This is just an option of course, you are not required to animate anything if you don't want to.
- **Can I sell the games I make with this program?**
  Yes, but good luck trying :P
- **Is there a limit to how big the game can be?**
  No, You can continue to add onto games using multiple event files and maps.
- **Why in gods name would you waste your time building this (even though its very cool)?**
  Because I enjoy the challenge of it all, not to mention that ascii games are awesome in general. I couldn't find an ASCII GAME MAKER on the internet… that had what I wanted, so I made one.
- **How hard is it to create a game with your program…?**

As easy as it can possibly be, IMO. If it were anymore straight forward and simple, It wouldn't be a challenge at all. It won't take long to understand how it all works.

Required Files

You will need to make sure that you have the required files need to construct and play your games. You may have downloaded this from a site other then my own. If this is the case, just make sure that you have what you need by checking the list below.

- **AQ-EDIT.EXE (editor)**
  This program will allow you to create all the files used by the engine.

- **GAME.DAT , QBPL.DAT, ASCP.DAT, EDIT.DAT**
  These files are actually programs. If you wish, you can rename the .DAT extention to .EXE if you wish to use this programs outside of the editor. The editor renames the files when using them to EXE. Be sure you leave the .DAT files there or the editor eill not be able to use them.

- **DATA (folder)**
  This folder contains all the external files used by the EDITOR and ENGINE. This folder must contain the following files or the engine will fail. Do not attempt to rename this folder. Doing so will cause an error.

  - **OBJECTS.DAT**
    This file contains all the WEAPONS, ARMOR, and SPELLS used in the game. This file has a special format, so be sure to follow the guidelines when creating or editing this file. One small typo can cause the engine to fail.
    (see OBJECTS.DAT section)

  - **ENEMY.DAT**
    This file contains all the ENEMIES / MONSTERS used in the game. This file has a special format, so be sure to follow the guidelines when creating or editing this file. One small typo can cause the engine to fail. (see ENEMY.DAT section)

  - **EVENT.TXT**
    This file contains all the EVENT SCRIPTING used in the game. This file has a special format, so be sure to follow the guidelines when creating or editing this file. One small typo can cause the engine to fail.
    (see EVENT.TXT section)

  - **WORLD_1.MAP**
    .MAP files are the actual maps that your character can explore during the game. The .MAP files are formatted in a very special way and are NOT meant to be edited in notepad. Use only the AQ-EDIT.EXE program to edit .MAP files to ensure the format is correct. (**World_1.map** is the default map that is loaded first when you begin a new game. Make sure that you do not rename or remove this map. All other maps can be named or renamed to whatever you like)

  - **MENU.ASC / GAME.ASC / BLANK.ASC**
    INTRO.ASC / HELP.ASC / CREDITS.ASC
    These files are all ascii graphics files created with ASCIIPAINT.EXE. There files are required by the game engine so DO NOT REMOVE or RENAME them. The MENU.ASC file is the first screen you see with the main game menu options listed. These options cannot be changed or renamed. (Intro, newgame, continue, help and credits are hard coded into the game engine. However you can edit the INTRO.ASC , HELP.ASC and CREDITS.ASC and BLANK.ASC to display what

ever you like.

- **PALETTE.DAT and SYSTEM1.FNT and CURSOR.IMG and WINDOW.IMG**
  These files are used only by the AQ-EDIT.EXE editor and do not need to included with your game UNLESS you plan to include the editor as well.

- **MUSIC.TXT**
  This file holds the default music for the game. Additional replacement music can be edited in this file, or loaded in from other files using the event modes.

Music and Sound effects

ASCIIQUEST uses the PLAY command to create sound and music for the game engine. Most of the default music and sound effects are stored in the MUSIC.TXT file. This file can be edited manually, but it is suggested that you use QBPLAY to create sound for your game. The QBPLAY will allow you to create / or edit music for you game and export the song into a PLAY command string. The play command string is formatted in a very special way. It is important that you understand how this format works to ensure that no errors occur.

Each note of your sound is formatted like such 'L16o1G+'. There are 7 characters required for each note… even if not all 7 characters are used. 'L8 o1G ' or 'P8 '. The first few characters set the duration of the note (L16 = 16th note L4 = quarter note). The next set of character represent the octave that is used. (o1 = first octave, o4= the fourth octave). The last set of characters hold the actual note to be played. (C+ = the note of C sharp).

Durations - L1 , L2, L4, L8, L16, L32, L64
Octaves - o1, o2, o3, o4, o5, o6
Notes – A, A+, B, C, C+, D, D+, E, F, F+, G, G+
Pauses – P1, P2, P4, P8, P16, P32, P64

NOTE: I haven't gotten around to streamlining the QBPLAY / ASCIIQUEST merge… For now QBPLAY will save or EXPORT to a file called SONG.TXT. From there you will have to manually copy and paste the line of music into the MUSIC.TXT file. I will get around to updating a special version of QBPLAY for ASCIIQUEST that will take care of the copying and pasting into the music.txt file for you. Sorry about the hassle.

Creating a MAP

(filename.MAP section)

A Map contains 21 tiles horizontally by 16 tile vertically. This provides the user with 336 tiles that must be created on a map for the game engine to properly run the game map. Each tile has 4 main properties. (**Tile COLOR , Tile ASCII , Tile BLOCK , Tile EVENT**)**.** By using the corresponding buttons provided in the editor, you can create the graphics for the map, what parts of the map you can walk on and which are not passable, and what tiles are linked to events. It is very important that you understand how this system works before you begin creating your first game. Lets go over the buttons that you see on the main screen.

View ASCII – This button will show the map ASCII graphics. Anything painted in this mode will cover all 4 propertires. (Tile COLOR , Tile ASCII , Tile BLOCK , Tile EVENT)

View BLOCK – This button will show the BLOCKED tiles in red. If you paint on this screen, Only the Block state will be drawn. Graphics and event scripts will not be drawn to the screen.

View SCRIPT – This button will show the tiles that contain scripts. Any tile that has event script attached to the tile will be labeled by the event ID number.

Tile ASCII – This button allows you to select an ASCII character to be painted onto the map screen.

Tile COLOR - This buttone will allow you to change the color of the ASCII character you want to paint onto the screen.

Tile BLOCK – This button allows you to switch whether or no the player can walk over a tile.

Tile EVENT – This button allows you to place and EVENT script onto the map. This will force the engine to perform an action when the player steps onto this tile.

ENCOUNTERS – This button will prompt you to enter The number of monsters you wish to randomly place on the map, and the ID# of the monster type you wish to use.

NEW – This will prompt the user for a new map name, Once entered, the current map will be saved as the new map name.

SAVE – This will save the current map to the specified filename.

EVENT ID – This button will allow you to jump to any event without scrolling.

MODE – This box holds the event mode “txt, heal, shop” etc…

CHAIN – The represents what event ID will follow.

PARAM 1 – Holds the first parameter of the event.

PARAM 2 – Holds the second parameter of the event.

PARAM 3 – Holds the third parameter of the event.

PARAM 4– Holds the fourth parameter of the event.

SUB – Displays the game OBJECTS.DAT file for easy reference.

ENEMY – Displays the ENEMY.DAT file for easy reference.

TEST – Allows you to test your game. Once finished testing, you return to the editor right where you left off.

TEST – This button will launch the game engine in DEBUG mode, showing you game variables and other important scripting information that will allow you to construct your games faster. Once you exit the game engine, you will be brought back to the editor.

Using the NPC engine

To place non interacting NPCs onto the map, Just enter a value greater than 0 for the Monster Count, and an ID# of 0. This will turn the enemies into friends that do not attack. They will however follow you around the map and block your path from time to time. It's a good idea to make the NPCS killable.

Scripting your ASCIIQUEST Game
(EVENT.TXT section)

The EVENT.TXT file contains all the scripting that runs the game engine. There are only 14 event modes that the engine will recognize. However, event scripts can be chained together to create complex events with multiple outcomes. Learning to use these event modes can be somewhat tricky at first, but once you understand the concept (or have any experience with game scripting) it will become easy in no time at all.

You may choose to edit the event.txt file using notepad, this is completely acceptable and recommended once you get a solid understanding of how the commands work. To start, you may wish to use the editor to create your event scripts. I have provided a few reference documents and worksheets to help you construct your games. These are vital and will help you a great deal when creating and evolving a plot.

If you open up the EVENT.TXT file, you will notice that it is formatted like so.

---------->>>> Beginning
0,"mode",0,"","","","",""

----------<<<< Ending

Note that there is a carriage return after the last line in the EVENT.TXT file. It is important that it remains there. If you are using the ASCIIQEDIT.EXE to script your events, this carriage return will automatically be inserted for you after each new event is added. If you are using notepad, make sure that you insert the carriage return manually or an error will occur.

Each line is formatted like this:
ID#, "mode" ,CHAIN#, "parameter 1" ," parameter 2" ," parameter 3" ," parameter 4"

- **ID#** is a unique number from 1 to 999. This number represents the ID associated with this particular event. It's a good idea to create a block of ID#'s for each map. This way it becomes easier to manage your scripts by knowing what blocks of ID#'s go with what map.
  (Example) : ID#'s 1 through 10 are scripts for map 1. (world_1.map)
  ID#'s 11 through 20 are scripts for map 2 (world_2.map) and so on…
  This is not required by any means, but will help you keep track of your scripting events.

- **MODE** is one of the event modes that can be used to cause the engine to do something specific. It can be a simple mode like **txt** which will display up to 4 lines of text on the screen, or **shop** which will let the user buy a weapon or armor object. Depending on what mode is used, the necessary parameters must follow accordingly. (See EVENT MODE Reference Sheet for details)

- **CHAIN#** is the ID# of another event to be called after the current event has been executed. This will allow you to chain together multiple events to construct an event series. By chaining a few simple events together, you can create complex events that will help to make you game more enjoyable and unique. Some commands like IFGATE and a few others have no need to CHAIN due to the way the command is implemented. If a CHAIN event ID# is supplied on one of these commands, the chain event make not occur.

  (Example):
  1, "txt", 2, "This is a test event", "It will chain to another", "When this event is complete", "Another event will be called"

2, “map”, “See how they work together”, “data/new.map”,10,10

- **PARAMETERs 1-4**
  These 4 parameters are different for each event mode. Make sure you know what parameters are needed for each event mode. These can be found on the EVENT MODE reference sheet.

ENEMY.DAT

The ENEMY.DAT contains a formatted list of all the monsters and creatures that a player can encounter while playing the game. It is important that the format of this file remain intact when editing in notepad. If the ASCIIQUEST.EXE program is used to add monsters to the ENEMY.DAT file, you need not worry about the formatting, The editor will take care of all that.

" ENEMY ID#**","MONSTER NAME","ATTACK","DEF","HP", "ASCII"**
1,"Blob",2,0,2,""

The first parameter is the ENEMY ID#, This is like all ID#;s used by the engine, it must be unique to the enemy.dat file. This number can be any number between (1-999). The 2nd parameter is the Monster Name. This should be no more than 10 characters max. The ATTACK parameter stores the value in which the monster uses to attack the player in combat. The DEFENSE parameter tells the game engine what defensive bonus the monster has. The HP parameter tells the engine how many hit points the monster has to lose before being destroyed by the player. The ASCII parameter tells the image what Ascii character to use when representing the monsters position on the screen.

NOTE: This first line of the ENEMY file should contain an ID# of 0 and the default values of the NPCs

Combat System.

The combat system is turn based. Who get the first swing is calculated randomly. The amount of gold received is a random number between 1 and the monster’s attack power. The Experienced gained is also the monsters Attack power. The engine is scripted to determine on a random basis whether or not the monsters will attack the player.

When in Testmode… The game engine will prevent the monsters from attacking you. So make sure your monsters aren’t to hard to destroy at low levels.

OBJECTS.DAT

The OBJECTS.DAT file is a very important file. This contains all of the WEAPONS, ARMOR and MAGIC that can be acquired during the game. The OBJECT.DAT file is formatted in a very special way, so it is important to double check your entries into this file. Each line of this file is formatted like this:

> **OBJECT ID#, NAME, ATTACKvalue, DEFENSEvalue, COST, RANGE**
> **(1-999) , "7 character or less ", (1-999),(1-999),(1-9999),(1-20)**
> **0,"dagger","1","0","23","0"**

By providing the game engine with the proper information, The player can access these objects by visiting Shops and Markets. But before you go adding items into the objects file, lets take a look at the required parameters. The first and most important parameter is the OBJECT ID#. This is a unique ID# that is attached to the OBJECT. When scripting shops, you will need to reference this OBJECT ID# to allow the game engine to provide the player the opportunity to purchase the object. This parameter is followed by the NAME parameter. The NAME tells the engine what to call this object. The NAME is used to display text on the character sheet and should be no more than 7 characters long. Anything longer than 7 characters may cause an error in the engine. The 3rd, 4th and 5th parameters change according to the type of object. One Parameter that remains consistent for all object types is the COST parameter. COST determines how much this item will cost if purchased from a shop. This can be a number from 1 - 9999

WEAPONS - The engine will recognize that an object is a weapon IF the ATTACKvalue is a value greater than 0. By setting the ATTACKvalue to 1, this tells the game engine not only is this object a weapon, but it has an attack power of 1. Weapons cannot have a DEFENSEvalue. If a DEFENSEvalue is present and the ATTACKvalue is greater than 0, The item is treated as WEAPON and its DEFENSEvalue is ignored by the engine. A valid ATTACKvalue can be any number between (0-999).

> **1,"dagger","1","0","23","0"**

ARMOR - The engine will recognize that an object is a weapon IF the ATTACKvalue is a value of 0 AND the DEFENSEvalue is greater than 0. By setting the DEFENSEvalue to 1, this tells the game engine not only is this object a armor, but it has an DEFENSEvalue of 1. Armor cannot have a ATTACKvalue. If a ATTACKvalue is present The item is treated as WEAPON and its DEFENSEvalue is ignored by the engine.
A valid DEFENSEvalue can be any number between (0-999).

> **1,"Coat","0","1","15","0"**

MAGIC - The engine will recognize that an object is a MAGIC object mainly because the script needed to call a magic object is different than the weapons and armor event scripts. Thus, the required parameters are different than the weapon and armor objects. The ATTACKvalue is the damage the magic spell causes. The DEFENSEvalue holds the COLOR of the spell, Not its defensive properties. The RANGE parameter tells the engine how far the spell can travel before running out of energy. The RANGE must be a value between (1 and 20). The MP needed by the player to cast the spell is equal to the ATTACKvalue. Therefore the more damage caused by the spell, the more MP's the player needs to cast it.

> **1,"Iceball","3","1","150","5"**

AsciiQuest Editor

EVENT MODE - Reference Sheet

	MODE	PARAM 1	PARAM 2	PARAM 3	PARAM 4
1	**txt**	“Text”	“Text”	“Text”	“Text”
2	**map**	“Welcome Text”	“data/name.map”	Load X position	Load Y position
3	**gold**	“Text”	“Text”	+ - Gold	
4	**shop**	“Welcome Text”	“Welcome Text”	Object ID#	
5	**heal**	“Welcome Text”	“Welcome Text”	Cost	
6	**hp**	“Text”	“Text”	+ - HP adjust	
7	**mp**	“Text”	“Text”	+ - MP adjust	
8	**learn**	“Text”	“Text”	Object ID#	
9	**market**	“Text”	“Text”	Cost	
10	**food**	Amount of food			
11	**gate**	Gate ID#	Gate Value		
12	**ifgate**	Gate ID#	If This Value	Goto This ID#	
13	**choice**	“Text”	“Text”	Goto YES ID#	Goto NO ID#
14	**get**	Type	Object ID#		
15	**lose**	Type			
16	**screen**	“Imagename.asc”	Pause # secs	xpos	ypos
17	**exper**	“Text”	“Text”	+ - EXP adjust	
18	**maxhp**	“Text”	“Text”	+ - maxHP	
19	**maxmp**	“Text”	“Text”	+ - maxMP	
20	**att**	“Text”	“Text”	+ - Attack	
21	**def**	“Text”	“Text”	+ - Defense	
22	**wis**	“Text”	“Text”	+ - Wisdom	
23	**elapse**	Number millisecs			
24	**song**	Filename.sng	songbank		
25	**enemy**	Id#	Xpos	Ypos	
26	**random**	From 1 to n#	Check number	Goto if <=	Goto if >
27	**chkstat**	stat	Check number	Goto if <=	Goto if >
28	**hidden**	ID#			
29	**ifkill**	Goto YES ID#	Goto NO ID#		
30	**sub**	Event file name	Event ID#		
31	**exitsub**				
32	**ctile**	Ascii character	Color	X cord	Y cord
33	**ptile**	Block (Y/N)	SCRIPT Event	X cord	Y cord

Modes are case sensitive. Make sure they are lowercase.

- **txt** – This MODE will display up to 4 lines of text on the screen. This is done 2 lines at a time. It is not required that all 4 “Text” parameters are used, and empty parameters will not be displayed. Make sure that there is NO space between the quotes or the engine will display the empty parameter. Each parameter of “text” can store a max of 35 characters.

  (Example): 1, “txt”, 0, “Hello world”, “This is my txt mode”, “it is easy to do”, “and fun”
  or….. 1, “txt”, 0, “Hello world”, “This is my txt mode”, “”, “”

- **map** – This MODE will display 1 line of optional text on the screen, Then will load a new map and place the character according to parameters 3 and 4 (the x and y).
  If the “text” parameter is omitted, No text will be displayed and you will jump right to the new map. Make sure when putting in your “name.map” that you include the “data/” before the map filename. Otherwise the engine will try to load the map from the root directory and this will cause an error.

  (Example): 1, “map”, 0, “Here is a map event”, “data/new.map”, “10”, “10”
  or….. 1, “map”, 0, “”, “data/new.map”, “10”, “10”

  You can also enter ‘X’ or ‘Y’ instead of a specific number value. This will automatically use the characters X or Y position as the new position when the map is loaded.

  (Example): 1, “map”, 0, “Here is a map event”, “data/new.map”, “**X**”, “10”
  or….. 1, “map”, 0, “Here is a map event”, “data/new.map”, “10”, “**Y**”

  This will take the current X or Y position of the character and use it as the new X or Y position for the character once the map is loaded. This is very handy when creating maps that allow the player to walk off the entire edge of the map instead of a small area.

- **gold** – This MODE will adjust the players gold amount. This can be used when purchasing information and items from NPC’s , Taxing, Bribery and any other instance a player is forced to give or receive gold for some reason or another. You can choose to display 2 lines of text, or leave them blank if your chaining events together.

  (Example): 1,”gold”, 0, ”Cool… I found a gold coin.”,”I think ill keep it.”,”+1”,””
  or 1,”gold”, 0, “”, “” , “+1”, “”
  and 1,”gold”, 0, “Yikes… a 15 gold tax?”, “I hate that evil king.” , “-15”, “”

- **shop** – This mode allows the player to purchase a weapon or piece of armor. The first 2 parameters hold the welcoming text. The 3rd parameter holds the OBJECT ID# which should reference an ID# stored in the OBJECTS.DAT file. If you do not want the item to be a weapons or armor, you can create an ID# in the OBJECTS.DAT file that has a value of 0 for both the ATT and DEF powers.

  (Example): 1,”shop”, 0, ”Welcome to my comic book store”,”Look here…”,”23”,””

- **heal** – This mode allows the player to be healed. The players HP and MP will be restored to the max. This mode will cost the player gold. If you want to heal the player for free, use the HP and MP modes. The 1st and 2nd parameters hold “welcome to the temple text”. The 3rd parameter holds the cost of the healing. The Engine is scripted to handle the transaction for you.

(Example): 1,”heal”, 0, ”Welcome to stone temple”,”let me heal you.”,”7”,” “

- **hp** – Use this mode to adjust the players hit points (HP). Parameters 1 and 2 are “Text” lines and parameter 3 holds the number to adjust. This number can be positive or negative.

  (Example): 1,”hp”, 0, ”Walking in this lake seems ”,”to heal you +1 HP”,”1”,””

- **mp** - Use this mode to adjust the players hit points (HP). Parameters 1 and 2 are “Text” lines and parameter 3 holds the number to adjust. This number can be positive or negative.

  (Example): 1,”mp”, 0, ”Walking in this lake seems ”,”restores magic +3 MP”,”3”,””

- **learn** – This mode will allow the player to upgrade or downgrade spell magic. Each spell is stored in the OBJECTS.DAT file and is referenced using its OBJECT ID#. The 1st and 2nd parameters store the “welcome to my magic shop text.” . The 3rd parameter holds the OBJECT ID# of the spell the player can purchase.

  (Example): 1,”learn”, 0, ”Welcome to my magic shop ”,”buy something”,”32”,””

- **market** – This mode will present the player the option of purchasing food. The first 2 parameters store the welcome text. The 3rd parameter stores the cost of 10 food units.

  (Example): 1,”market”, 0, ”We have fresh fruit. ”,”We also have nuts.”,”2”,””

- **food -** This mode will present the player with free food. Parameter 1 holds the amount of food the player will get when this event is triggered.

  (Example): 1,”food”, 0, ”3 ”,””,””,””

- **gate** – This is a very special mode. It does not display any text, but it very useful when creating complex plot triggers and gates. By using this command, you can tell the engine that the player has completed a task of some sort. “Found a key”, or “pulled the lever”. These task gates are saved when the player saves their game. This ensures that the plot scripting can continue along as the player saves and loads their games.
  The 1st parameter is the ID# of the gate. The 2nd parameter is the value the gate will have. ID# can be a number from 1 to 100 and the value can be a number from 1 to 32000. This provides the creator with nearly an unlimited amount of gates to control the plot and game environment.

  (Example): 1,”gate”, 0, ”1” , ”5” , ”” , ””

  You can also adjust the gate value using the ‘+’ or ‘-‘ operators. If a ‘+’ is inserted into parameter 2 instead of a number, the gate value is incremented by 1 (gate = gate + 1). The same goes for the ‘-‘ operator. (gate = gate – 1).

  (Example): 1,”gate”, 0, ”1” , ”+” , ”” , ””

- **ifgate** – This mode is the counterpart to the gate mode. Ifgate allows you to check if a particular gate has a particular value… and if it does, what event ID# to load as a result.

  (Example): 1,"ifgate", 0, "1" , "5" , "10" , ""
  1,"txt", 0, "yep gate 1 has a value" , "of the number 5" , "" , ""

- **choice** – This mode (although somewhat redundant) can make easy scripting of a simple Yes or No decision. The player is presented with the ability to input 'y' for yes and 'n' for no. Each choice is connected to an event ID#. Parameter 1 and 2 store the choice text. Parameter 3 holds the YES ID# and parameter 4 carries the NO ID#.

  (Example): 1,"choice", 0, "Open the door?" , "Yes or No? (y/n)" , "10" , "20"

- **get** – This mode gives the player a free object. The 1st parameter tells whether the object is a **"weapon"** or an **"armor".** The second parameter holds the OBJECT ID#.

  (Example): 1,"get", 0, "weapon" , "3" , "" , ""

- **lose** – This mode forces the player to drop a "weapon" , "armor" or "magic". Once the object is dropped, it cannot be returned to the player. The object must be re-aquired using event-scripting.The 1st and only parameter required for this mode is the TYPE. The TYPE can be 1 of the 3, a "weapon" , "armor" or "magic".

  (Example): 1,"lose", 0, "magic" , "" , "" , ""

- **screen** – This mode will load an .ASC file onto the screen at the given X and Y parameters. This file can cover the whole screen or just a small portion of it. The .ASC is created using ASCIIPAINT.EXE. The mode operates as follows. Parameter 1 stores the "filename.asc". parameter 2 holds the number of seconds to pause the display. Any number of 99 will result in the user having to press a key to continue. The 3rd and 4th parameters hold the X and Y position of the ASC image.

  (Example): 1,"screen", 0, "data/ending.asc" , "15" , "1" , "1"

- **exper** – This mode will adjust the players experience points. This can be a positive or negative number between –32000 and +32000. This mode will perform a check to see if the character will raise a level after the experience has been adjusted. You can not for a character to drop a level… But you can force them to re-earn lost experience before raising levels in the future. The first 2 parameters hold text for this mode, the third parameter holds the adjust value.

  (Example): 1,"exper", 0, "You gain some experience" , "15 points." , "15" , ""

- **maxhp –** This mode will adjust the players max hit points. This should be a number no greater than 999. The first 2 parameters hold text, the 3rd parameter holds the adjustment value.

(Example): 1,”maxhp”, 0, ”Ive adjusted your stats.” , ”by 5 points.” , ”5” , ””

- **maxmp –** This mode will adjust the players max magic points. This should be a number no greater than 999. The first 2 parameters hold text, the 3rd parameter holds the adjustment value.

  (Example): 1,”maxmp”, 0, ”Ive adjusted your stats.” , ”by 5 points.” , ”5” , ””

- **att –** This mode will adjust the players attack power points. This should be a number no greater than 999. The first 2 parameters hold text, the 3rd parameter holds the adjustment value.

  (Example): 1,”att”, 0, ”Ive adjusted your stats.” , ”by 5 points.” , ”5” , ””

- **def –** This mode will adjust the players defensive power points. This should be a number no greater than 999. The first 2 parameters hold text, the 3rd parameter holds the adjustment value.

  (Example): 1,”def”, 0, ”Ive adjusted your stats.” , ”by 5 points.” , ”5” , ””

- **wis –** This mode will adjust the players wisdom power points. This should be a number no greater than 999. The first 2 parameters hold text, the 3rd parameter holds the adjustment value.

  (Example): 1,”wis”, 0, ”Ive adjusted your stats.” , ”by 5 points.” , ”5” , ””

- **enemy–** This mode will place an enemy on the screen. This command is great for creating boss enemies or forcing a player to defete an enemy before exiting a level. The first parameter holds the monster ID#, the second parameter holds the X position of the new monster, and the 3rd parameter holds the Y position.

  (Example): 1,”enemy”, 0, ”1” , ”10” , ”10” , ””

- **elapse –** This mode will pause the game for a number of milliseconds. This can be used to animate characters on the screen. This first and only parameter holds the number of milliseconds to pause the game.

  (Example): 1,”elapse”, 0, ”1.5” , ”” , ”” , ””

- **song –** This mode will change the loaded song for an ingame action. There are 6 song modes that can be switched. They are (gamesong : introsong : enemysong : attacksong : victorysong : levelupsong).
  This first parameter holds the FILENAME.SNG of the song to be loaded. The second parameter holds where the song will be stored (gamesong : introsong : enemysong : attacksong : victorysong : levelupsong). See MUSIC section for more information about song file formats.

  (Example): 1,”song”, 0, ”newsong.sng” , ”enemysong” , ”” , ””

- **random** – This mode provides you with the ability to script random events into the game. Guessing games and casinos and more… The first parameter holds the limit of the random number. (0 to parameter 1). The second parameter holds the check value. The check value acts as a divider for parameter 1. Parameter 3 tells the game engine what

event id# to jump to if the value of parameter 2 is less than or equal to the random number. Parameter 4 tells the game engine what event id# to jump to if the value of parameter 2 is greater than the random number. Here is a simple breakdown of how this works.

(Example): 1,”random”, 0, ”100” , ”10” , ”25” , ”45”

This statement picks a random number from 0 to 100, if the number is less than or equal to 10 goto ID# 25, if the random number is greater than 10, goto 45

- **chkstat** – This mode will let you check to see if a particular stat is less than or equal to, or greater than the specified number. Parameter 1 holds what stat you wish to check. The valid check stats are (**maxhp, maxmp, lvl, att, def, wis, gold**). The second parameter holds the value to be checked. The 3rd parameter holds the event ID# that the game engine will jump to if parameter 2 is greater then the specified stat (parameter 1). The 4th parameter holds the event ID# that the game engine will jump to if the stat is less than or equal the specified value of parameter 2.

  (Example): 1,”chkstat”, 0, ”lvl” , ”10” , ”13” , ”14”

  This statement check to see if the players level (lvl) is greater than 10, if it is less than or equal to 10, it will jump to event 13, if it is greater than 10 it jumps to event 14.

- **hidden** – This mode will place a hidden event onto a map. This hidden event can be ‘searched’ for by the player pressing the ‘x’ key when standing on a tile. Parameter 1 holds the event ID# the engine will jump to if the player searches on the tile where the hidden event was placed.

  (Example): 1,”hidden”, 0, ”23” , ”” , ”” , ””

- **ifkill** This mode checks to see if all monsters on the map have been destroyed. If they have, the engine will jump to the event ID# stored in parameter 1. If not, the event will jump to the event ID# stored in parameter 2. If parameter 2 = 0 or is left blank, the engine will do nothing.

  (Example): 1,”ifkill”, 0, ”1” , ”2” , ”” , ””
  or 1,”ifkill”, 0, ”1” , ”” , ”” , ””

- **chkstat** – This mode provides you with the ability to check a players stat value, and jump to an event ID# according to whether or not the checkvalue is (greater than or less than or equal to). There are 6 stats that you can check for. (lvl, maxhp, maxmp, att, def, wis). The first parameter of this statement tells the engine what stat to check. The second parameter holds the checkvalue. Parameter 3 tells the game engine what event id# to jump to if the value of parameter 2 is less than or equal to the selected stat. Parameter 4 tells the game engine what event id# to jump to if the value of parameter 2 is greater than the selected stat. Here is a simple breakdown of how this works.

  (Example): 1,"chkstat" ,0, "lvl" , "10" , "25" , "45"

  This statement checks to see if the characters level is less than or equal to 10 if it is the engine will jump to event ID# 25. If the characters level is greater than 10, the engine will jump to event ID# 45

- **hidden** – This will place a hidden event onto a map. When the character walks over this tile and presses the search button 'x' , the event ID number (parameter 1) will be called. There is no limit to the number of hidden events allowed per map.

  (Example): 1,"hidden" ,0, "23" , "" , "" , ""

- **sub** – This will force the engine to use a specified event file. This file is just like the EVENT.TXT file, however you can separate parts of your event scripts into smaller , more manageable chunks. For example, you may wish to create a sub for each map, thus loading a new events file each time you load a new map. Once this command is called, all scripts will be executed from this newly loaded file until the exitsub command is given. The first parameter holds the value of the event script file name. The 2nd parameter tells the engine where to start reading by executing the event id# provided.

  (Example): 1,"sub" ,0, "casino.evt" , "1" , "" , ""
  or 1,"sub" ,0, "casino.whatever_you_like" , "1" , "" , ""

  This statement will tell the engine to open the 'casino.evt' file and start by reading event # 1.

- **exitsub** – This will force the engine to exit or unload the previously loaded event file, and revert to the 'EVENT.TXT' file. No parameters are required.

  (Example): 1,"exitsub" ,0, "" , "" , "" , ""

- **ctile -** This command will change a tile on the map. This will only change the color and ascii character to display, the BLOCK state and EVENT attributes of the tile are left intact. This is a great command for placing objects on the maps that you want the player to pick up.

  (Example): 1,”ctile” ,0, “65” , ”15” , ”10” , ”10”

- **ptile -** This command will change a tile on the map. This will only change the BLOCK state and EVENT attributes, the color and ascii character of the tile are left intact. This is a great command for blocking and scripting on the maps.

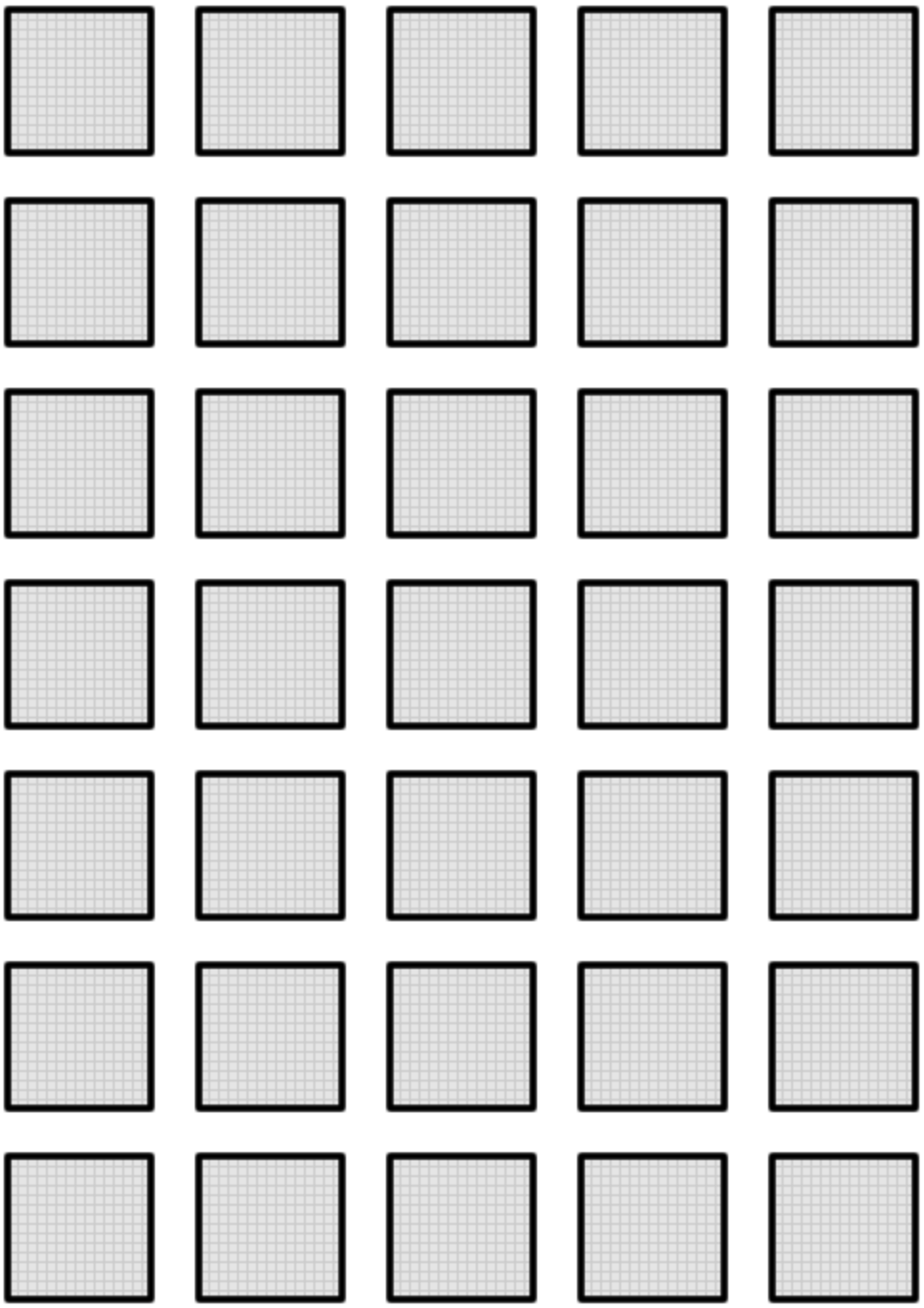
  (Example): 1,”ptile” ,0, “Y” , ”0” , ”10” , ”10”
  This will change the block attribute of tile 10, 10 to ‘Yes”
  1,”ptile” ,0, “N” , ”123” , ”10” , ”10”
  This will change the tile 10,10 block attribute to ‘No’ AND set the tile to run script ID# 123 when the player passes onto the tile.

AsciiQuest Editor

GATE LOG - Reference Sheet

Gate#	Value	Script#	Notes

MAPS - Tracking Sheet
Use this sheet to help you plots your maps.



ENEMY - Tracking Sheet

Use this sheet to help you log what enemies you have created and used

ID#	MONSTER	ATTACK	DEFENSE	HP	ASCII	Maps Used on

OBJECT- Tracking Sheet

Use this sheet to help you log what objects you have created and used.

ID#	NAME	ATTACK	DEFENSE	COST	RANGE	EVENTS