

VAPP

Visual and Auditory Presentation Package for Intel ISA computers

Usage and Reference manual

This software is licensed for research purposes only, and is not for use in clinical applications.

1. Introduction	2
1.1 Execution environment	3
1.1.1 Operating systems	3
1.1.2 System requirements	3
2. Installation	4
2.1 Before you begin	4
2.2 Installing VAPP	4
2.2.1 AUTOEXEC.BAT	4
2.2.2 Sound card driver	4
2.2.3 SciTech Display Doctor	5
3. Usage	6
3.1 Invocation	6
3.2 Startup Options	6
3.2.1 VAPP.CFG	6
3.2.2 Command line options	6
3.3 File locations	11
4. Operation	11
4.1 Codes	12
4.2 Errors	12
4.2.1 Compilation Errors	13
4.2.2 Execution Errors	13
5. VGA Graphics Environment	13
5.1 Resolution and Coordinates	13
5.2 Colours and Palettes	14
5.2.1 Bitmaps, background images, and palettes	14
6. Sound playback environment	15
6.1 The WAV format	15
6.2 Mono and Stereo files	15
7. Display, Timing, and Buffering Algorithms	16

7.1 Frame Buffering.....	16
7.2 Timing Considerations.....	16
8. Stim Files	17
8.1 General	17
8.1.1 String arguments.....	18
8.1.2 Comments	18
8.2 Mandatory Arguments	18
8.2.1 SOA	18
8.2.2 Presentation Duration	19
8.2.3 Code	19
8.2.4 Basic Stimulus	19
8.3 Options	20
8.3.1 Xoff and Yoff.....	21
8.3.2 Colour	21
8.3.3 Label Origin.....	21
8.3.4 Font.....	22
8.3.5 Parallel output	22
8.3.6 Volume.....	23
8.3.7 Balance	23
8.3.8 Continuation.....	23
8.4 Stimulus Types	24
8.4.1 Text.....	24
8.4.2 Image.....	24
8.4.3 Pgi	24
8.4.4 Sound	24
9. Pgi Files	25
9.1 General	25
9.2 Pgi Plotting Environment	25
9.3 Commands	26
9.3.1 State Commands	26
9.3.1.1 setfgcolour pixelvalue	27
9.3.1.2 setbgcolour pixelvalue	27
9.3.1.3 setlinetype mask.....	27

9.3.1.4	setdrawmode drawmode.....	27
9.3.1.5	selectfp fillpatternindex	28
9.3.1.6	setlblorig labelorigin	28
9.3.2	lineto xc yc	28
9.3.3	rlineto xc yc	28
9.3.4	moveto xc yc	28
9.3.5	rmoveto xc yc.....	28
9.3.6	orect nxpix nypix	28
9.3.7	frect nxpix nypix	29
9.3.8	image bmpfile.....	29
9.3.9	label "text string"	29
9.3.10	font fontfile	29
9.3.11	pgi pgifile	29
9.3.12	Polygon Commands.....	30
9.3.12.1	Startpgon and Endpgon.....	30
9.3.12.2	outlinepgon rx ry	30
9.3.12.3	routlinepgon rx ry.....	30
9.3.12.4	fillpgon rx ry	31
9.3.12.5	rfillpgon rx ry.....	31
10.	The log files	31
10.1	Name and location.....	31
10.2	Contents.....	32
10.2.1	LOG.....	32
10.2.2	CSV	32
10.2.3	Error messages	33
10.2.4	Stimulus times	33
11.	See Also	33
12.	Possible Future Enhancements	34
12.1	Suport for other image file formats.....	34
12.2	Motion picture stimulus presentation.....	34
12.3	Alteration of default options within the stim file.....	34
13.	Appendix 1: Default VGA Colour-mapping Palette.....	35

14. Appendix 2: Label Origin Positioning System _____ 36

VAPP

Visual and Auditory Presentation Package for Intel ISA computers

Usage and Reference manual

1. INTRODUCTION

VAPP is a program that displays visual images and plays pre-recorded sounds on an Intel ISA computer for use in event-related potential (ERP), perceptual, or psychophysical experiments. The visual images, termed stimuli, are presented with carefully controlled durations and stimulus onset asynchrony times (SOA) on a colour monitor. Synchronization codes (called event codes) can be sent out the serial and/or parallel port to signal the onset of presentation for each stimulus for use in signal averaging, measurement of reaction times, or other analytic procedures. VAPP also supports the connection of a custom response device to the modem-control lines in a serial port for recording patient responses to stimuli. Written in the “C++” programming language, VAPP runs in an MS-DOS operating system version 6.22 or earlier and is compiled using version 11.0a of the Watcom C++ compiler.

The specifications for the stimuli, their durations, SOAs, associated codes, and other parameters are contained in a text-only **stim file**. The stim file can be created using any text editor (e.g. **VI, Notepad, or Edit**), or by a special purpose program designed for the specific experimental requirements. Often they are created automatically by the program SG (Scenario Generator); for which a Use and Reference manual is available.

VAPP is quite flexible and can be used to present visual and auditory stimuli for a wide range of experiments. Considerable effort was spent attempting to anticipate possible experimental stimulation requirements during its design and development. Some of its features include:

- Simple and straightforward specification of multiple types of images using ASCII text files.
- Ability to import images from other graphics programs, e.g. Windows Paintbrush.
- Ability to employ a standing background image or a solid colored background between and during presentation of stimuli.
- Careful control of stimulus timing using multiple video frame buffers.
- Fast execution of graphics primitives by virtue of:
 - precompilation of the stim file before presentation,

- memory-resident image information,
- exploiting extended memory to reduce disk access.
- Simple use of .WAV sound files.
- Low cost.

VAPP is the successor to VSPRES, SBVSPRES, VVSP, and SHOWSTIM, and incorporates many of their functions.

1.1 Execution environment

VAPP is a generic SVGA version of VVSP with an image resolution of 640 horizontal by 480 vertical pixels, displaying 256 simultaneous colours out of a palette of 262,000. To implement stable, reliable timing on a generic VGA card in the DOS operating system requires detailed foreknowledge of the performance of the host computer through a broad spectrum of processing tasks. To get this performance information, VAPP profiles your computer's operation at startup time. It is possible for VAPP to declare a particular computer unfit for stimulus presentation if performance is inadequate.

1.1.1 Operating systems

VAPP is intended to run in a DOS environment. This rather severe restriction is imposed by the real-time response requirements of producing stable timing. None of Windows 3.x, Windows 95, or Windows 98 is capable of real-time response and cannot produce system timer services with the required resolution. Other operating systems (namely Windows NT, QNX, and the real time unix variants, among others) are capable of delivering the system timing required, but one of the objectives of VAPP is that it be easily distributable, which precludes requiring the user to install another operating system.

For users running Windows 3.x, you need only leave Windows to be in a true DOS environment from which VAPP may be run. If you are using Windows 95, 98, or NT, *you will need to set up a dual-boot configuration which allows you to boot the machine directly into DOS version 6.22 or earlier.* Note that the DOS box under Windows is not acceptable for running VAPP, nor is DOS 7.0 that comes with Windows 95. **Though VAPP will execute in DOS 7.0, precise timing cannot be guaranteed.**

1.1.2 System requirements

VAPP is designed to run on a standard VESA compliant SVGA card with hardware acceleration capability. The software will take advantage of installed hardware to maximize the speed of presentation, so the ability to display high speed sequences of stimuli will depend in part on the type of computer and the installed options.

Required	Pentium or higher processor, 233 Mhz or faster
	16 Mb Extended memory
	MS-DOS 6.22

Hardware accelerated Video Card

Presently, the only video cards on which VAPP has been thoroughly tested and is guaranteed to perform to specification are the ATI family of Mach 64 based devices, including the "Pro Turbo", and "Xpert.." series. AGP is fully supported through the Display Doctor. You may get satisfactory performance from other video cards or not depending on their implementation of the VESA VBE accelerator standard. The full list of video cards that are supported by the Scitech Display Doctor is on the Scitech web site at <http://www.scitechsoft.com>

2. INSTALLATION

2.1 Before you begin...

The most important starting point is to set up a DOS 6.22 environment on your computer. If your hard drive has been formatted with the FAT32 or NTFS filesystems, you will need to create a FAT16 partition and install DOS to that. If you are starting from a bare machine, you can do this with FDISK. If you want to mix and match partition types, or want to preserve existing data on the hard disk, you will need to use a third party partitioning program like the Partition Magic utility. If you have never done this before, you should enlist the help of an experienced software technician.

2.2 Installing VAPP

VAPP is distributed as a ZIP archive. To install it on your computer, you should create a directory in which you want to place VAPP, change to that directory, and run

```
PKUNZIP -d A:VAPP.ZIP
```

This will place the executable in the directory you created, and support files in subdirectories under that directory. The support files include over 300 font files, sample sounds and images, and a sample stim file to get you started.

2.2.1 AUTOEXEC.BAT

You should add the directory where VAPP.EXE was installed to your PATH, statement in your AUTOEXEC.BAT file. This is needed if you intend to keep your stim files separate from the EXE installation directory, which is recommended. When editing the AUTOEXEC.BAT file, be sure you are editing the correct one –Win 95/98 renames the multi-boot DOS AUTOEXEC.BAT file to AUTOEXEC.DOS while it is running. The way to be sure you are editing the right one is to boot the machine into the DOS 6.22 environment before proceeding with any other installation.

2.2.2 Sound card driver

Once DOS is installed, you should install the Sound Blaster drivers if this hasn't already been done.

You can check to see if the installation has already been done by looking in your AUTOEXEC.BAT and seeing if there is a line

SET BLASTER=

followed by numerical settings. If that line is not in your autoexec, you must install a Sound Blaster driver from within DOS.

If you have a DOS installation program that was provided with your sound card, use that and then proceed to the Display Doctor install. If you don't have any software provided with the sound card, you should try the Sound Blaster install diskette provided with your copy of VAPP.

If the installation does not succeed (look in the autoexec for the BLASTER= line), and you have a plug 'n play sound blaster, you may need the Creative PnP configuration manager, CTCM.EXE. This allows DOS programs to work with PnP devices. I have provided a copy of this utility, though it is also available from Creative Labs at: <ftp://ftp.creaf.com/pub/creative/patches/ctcmbbs.exe>

So if the installation fails, install the software on the CTCM diskette and then re-run the SB16 installation. If the latter still fails, then it is likely that your sound card is not SB16 compatible. Please contact me if you have difficulty with sound installation.

2.2.3 *SciTech Display Doctor*

Display Doctor is a hardware accelerator driver which loads at boot time that will handle the interface to your video card. It requires no further operation beyond installation. It will have been provided for you in one of two forms, either as a single diskette DOS based installation, or as a shrink-wrapped complete package from Scitech.

DOS Installation:

Run the A:Setup program from within DOS. When it has completed, you must still manually enter a line in your AUTOEXEC.BAT to launch the application UNIVBE.EXE, using a complete path to wherever you installed the display doctor, i.e.: C:\SDD\UNIVBE.EXE.

Shrink-wrapped package from Scitech:

The installation program for this version runs under Windows. VAPP, of course, runs under DOS. In Win95/98, if you run the install program, it will make modifications to your AUTOEXEC.BAT to load the driver at startup. The problem with this is that the changes are made to the wrong file – as mentioned above, the AUTOEXEC file used during the DOS boot is renamed to AUTOEXEC.DOS while Windows95 is running. You should run the installation normally under whichever version of Windows you have installed, then use an editor to copy the line that loads UNIVBE.EXE from the file AUTOEXEC.BAT to AUTOEXEC.DOS. You will know everything is functioning correctly when you reboot in DOS and see the Scitech load message while your machine is booting.

If you have not received the Display Doctor, you will need to obtain it from Scitech at 505 Wall St, Chico CA, 95928-5624, (916) 894-8400, www.scitechsoft.com. The product can be purchased online, and at the time this manual was written cost less than \$50.

3. USAGE

VAPP is easy to use once you have prepared a stim file. You must ensure that the DOS PATH is set so that VAPP can be invoked from any directory which contains stim files. You must do this manually by editing the PATH= line in your AUTOEXEC.BAT file.

3.1 Invocation

VAPP is invoked from DOS with:

```
vapp stimfile [options]
```

where stimfile is the name of the desired stimulus description file. Stim files have no default extension, so you are free to use whatever you want. You must specify the extension when VAPP is invoked.

3.2 Startup Options

There are a number of settable options which are specified at startup time that control how VAPP behaves during the stim presentation session. They each have a default value, can be specified on the command line when VAPP is started, or can be put in the VAPP.CFG configuration file.

3.2.1 VAPP.CFG

A configuration file, called `VAPP.CFG` will be searched for first in the current directory, and then in the directory in which `VAPP.EXE` is located. If a file is found in the current directory, it is used instead of the "global" file saved with the executable. In this way, options may be specified for a group of stimulus runs by keeping them together in a directory with a common CFG file.

3.2.2 Command line options

Once the configuration file has been read in, the command line is checked for options following the name of the stim file. The syntax for specifying the options is the same on the command line as in the .CFG file, namely `tag=value`. Also, for any on/off values that default to 0(off), you may turn them on by simply naming them and omitting the =1.

Any options specified on the command line will supercede corresponding entries in the .CFG file.

Option	Default	Description
<code>bgc= bgcolor</code>	0	Use <code>bgcolor</code> as default bg color.
<code>csvremove=0 1</code>	0	Include stim <i>removal</i> in CSV file
<code>dur=delta_dur</code>	0	Add delta_dur to all durations.
<code>echo=0 1</code>	0	Echoes the input devices to the output port. 1 to turn on, 0 for off, default is off.
<code>escapequits=0 1</code>	1	Run can be terminated by the escape key.

exitonerror=0 1	1	Terminates a stim run on any error, including late stim presentation
font= fontname	""	Use fontname as default font.
fontsz= size	5	Sets the size of the font for stimulus text.
SOA= delta_SOA	0	Add delta_SOA to all SOAs.
logfile=fname	yymmddnn	Uses fname as the name of the log file to be output.
luminance= brightness	10	Sets the brightness for drawn text and pgi graphics
nocsv=0 1	0	Do not write the .CSV file
nolog=0 1	0	Do not write the .LOG file
nosound=0 1	0	Disable sound card.
pal= palette	""	Use palette as default color map
responseport=[0 1 2 3 4]	0	Monitor for response device on COM port specified, 0 for none.
sbg= bg_img	""	Use bg_img as a standing background image.
setcodes= key_map	""	Establish the codes to be associated with keystrokes
setoutport=[0 1 2 3 4]	0	Sets the COM port to which codes are written, 0 for none.
setinput=[k ;1; 2 ;g]	'k'	Sets the input device(s) which will be monitored for input during the presentation run.
spause=0 1	0	Enable selective pauses. 1 to turn on, 0 for off, default is off.
textpalette=0 1	0	Optimize the sbg background image's palette for display with text and line drawing stimuli.
timerfix=0 1	1	Lock the VAPP clock to the 8259 timer in the PC on Pentium and higher processors.
waitforkey=0 1	1	Delay first stim presentation until keyboard key pressed.
waitforport=0 1	0	Delay first stim presentation until pin 1 on LPT1 toggles low->high.
xres=640 800 1024	640	Select screen resolution, with yres.
yres=480 600 768	480	Select screen resolution, with xres.

Here is a brief explanation of the use and effect of each option:

bgc Normally, the default background color that is used to erase previously drawn stimuli and which is used during drawing operations is black (0). By using this option, one can select an alternative background color **bgcolor** as a number between 0 and 255 (inclusive). The colour that this integer denotes is

	determined by the palette in use. Appendix 1 contains a list of the first sixteen colours (0-15) in the default palette. However, one can employ editvpal to create a custom palette along with the -pal option to obtain any background color desired. Note that before any stimuli are drawn the entire display is set to the background colour.
csvremove	The CSV output file normally contains only the presentation of a stimulus. If you are interested in the removal as well, then specify this flag in the CFG file or on the command line.
dur	one can alter the duration of all stimuli in a stim file using this option. One argument is required, delta_dur , which is the number of milliseconds to increase (if delta_dur is positive) or decrease (if delta_dur is negative) all durations. Remember that all temporal extents are transformed to the quantum units of frames, or 16.66 msec at 60 Hz refresh rate. No duration can be reduced below 1 frame; all such requests will be forced to 1 frame.
echo	Turns on or off the echoing of codes out the output port (selected with setoutputport) for keystrokes. The choices of which byte is sent out the port for particular keystrokes is established with setcodes below.
escapequits	When on (1), the stim run can be interrupted by hitting the escape key.
exitonerror	To prevent gathering possibly invalid data, a stim run is always terminated on any error or if a stimulus cannot be presented on time. You may relax this requirement by setting this flag to 0.
font	The default font for text can be selected by employing this option with the name of the desired font as fontname . VAPP supports standard Windows .FON files for fonts.
fontsz	This option is used to scale the text drawn as part of a stimulus image. A number is specified between 1 and 100. The fontsz divided by 10 is the scaling factor applied to the text. It only affects vector fonts.
SOA	as with the duration of stimuli, one can alter the SOAs of all stimuli in a stim file using this option. One argument is required, delta_SOA , which is the number of milliseconds to increase (if delta_SOA is positive) or decrease (if delta_SOA is negative) all SOAs. Again, remember that all temporal extents are transformed to the quantum units of frames, or 16.66 msec at a 60 Hz refresh rate. No SOA can be reduced below the duration of the preceding stimulus and all such requests result in the prior duration being used as the SOA. The original documentaion, and the initial implementation of VAPP refer to the SOA (Stimulus Onset Asynchrony) as SOA (Inter Stimulus Interval) throughout, a misnomer inherited from the VVSP predecessor. Though the documentation has been corrected to

	use SOA, the references within the program itself remain in place so people wouldn't have to alter any existing configurations.
logfile	Ordinarily, two log files are produced, one with a .LOG extension which contains a verbose human-readable script describing the results of the stim run, and one with the .CSV extension, ready for import into a spreadsheet package. The default base name is made up of the current date plus a two-digit index of the run in that day. You may override this name by specifying the logfile name in the configuration file or on the command line. Do not include a period or extension.
luminance	This determines the brightness of text drawn on the display. It is a number between 0 and 10, with 0 being black and 10 being the brightest the display can generate. This applies to drawn text and pgi graphics, but not to .bmp images contained in pgi graphics.
nocsv	By default, a CSV file is written, whose content is detailed below. If you do not want the CSV file, you can disable it by turning this option on.
nolog	As above, for the .LOG file.
nosound	Disables access to the sound card. Set this option =1 (or just name it) if you do not have a sound card installed in your computer, or if you want to suppress sound playback on computers which have sound cards installed.
pal	The argument to this option, palette , is used as the default colour mapping palette during stimulus presentation. Palette is the name of a VGA colour palette file (probably generated using editvpal).
responseport	The response device button up and down events appear in the log file with the exact time that the transition took place (to within about a microsecond). Set the response port to the com number, ie responseport=1 or responseport=2. Currently only COM1 and COM2 are supported. If the response device is not present, set responseport=0 (this is the default value).
sbg	This option instructs VAPP to employ the image in the image file <u>bg_img</u> as the standing background on which stimuli are superimposed, rather than a solid background colour. The use of a standing background image reduces the number of drawing pages by one; hence the timing constraints are more rigorous and certain sequences of stimuli that can be drawn without this option may not be able to be displayed when it is in effect. The background image in <u>bg_img</u> is normally a full screen (640x480) image. It is possible, however, to employ smaller images; these will be centered on the screen with the border set to the background colour.
setcodes	Establishes the mapping between keys on the keyboard and the

bytes to send out the output port when the **echo** option is turned on. The map is specified as a sequence of individual mappings of the form *character:number* where the character is a single quote enclosed character or escaped numeric value for unprintable characters, and the number is a decimal number from 0 to 255.

```
setcodes='j':59,'\0x20':21,'\ ':55
```

would see a byte value of 59 output when the 'j' key on the keyboard was pressed, 21 for the space bar (which happens to have ascii value 32, or 0x20, and 55 for the single quote character itself.

setoutput	Sets the output device to which codes are written. Only one may be selected at a time. The four serial ports are the only devices supported.
setinput	Sets the input device(s) which will be monitored for input during the presentation run. Monitored inputs have all of their state changes echoed to the log file with the exact time in the run. To specify more than one device for monitoring, separate the names with commas
spause	Normally one can request a pause in the presentation of images at the end of any stimulus in the stim file. If one employs this option, both keyboard pauses and hard pauses are only allowed at the end of specially-marked entries in the stim file.
textpalette	When selected, the background image's colors will be mapped into the default palette, the one used for line and text drawing. This will make the sbg image colors look fairly correct when displayed under text. Without this option, the intensity values in the sbg image will use whatever palette belongs with the current stim, without translation. This will result in distorted colors when drawn with text, but will also mean exact color matching when the stim is a bitmap image with the same palette. For more details, see section 5.2, "Colours and Palettes".
timerfix	The internal VAPP clock uses a special CPU instruction to get very precise timing. The absolute accuracy of that clock, however, can depend on your exact CPU speed, which varies with different computers. By default, the VAPP clock is synchronized to the 8259 clock chip in the PC so the accuracy is the same as the time-of-day clock.
waitforkey	VAPP waits with the background image in place (if any) until a keyboard key is pressed, otherwise it launches immediately into stimulus presentation as soon as it is ready.
waitforport	Delay first stim presentation until pin 1 on LPT1 toggles low->high. This is used to synchronize the start of a stim run with an external device, originally a GE MRI machine. When waitforport is selected, waitforkey is disabled. This works well with a TTL input that idles high, and sends a negative pulse as

a trigger.

xres,yres Together, these select the screen resolution to be used. The resolutions available depend on the video card installed in the your computer, along with the level of support for that hardware available in the version of the Display Doctor that you are using. They need to be used in standard combinations to work correctly, like xres=800 and yres=600, or xres=1024 and yres=786. These options are still specified on separate lines in the cfg file.

Any combination of options is permitted; if more than one is employed they can be supplied in any order.

3.3 File locations

The stimulus files, including the main stim run named on the VAPP command line and also any included PGI files, are only searched for in the current directory. This means that when you are designing an experiment, you should make a subdirectory for that experiment which contains all of the stim files and invoke VAPP from that directory. This need not (and probably should not) be the directory in which VAPP.EXE is installed, though you must add the directory containing VAPP.EXE to the DOS PATH in your AUTOEXEC.BAT to put your stimfiles elsewhere.

Associated data files used during stimulus runs (.BMP, .PCX, .JPG, .WAV, .FNT, .PGI and any others) are searched for first in the current directory, then in the subdirectory below the current directory with the name appropriate to the file type:

.WAV sound files	SOUNDS
.FNT font files	FONTS
.BMP image files	BITMAPS
.PCX image files	BITMAPS
.JPG image files	BITMAPS

If the file is not found in the current directory or the named subdirectory under the current directory, then a final attempt is made to find the file in the named subdirectory under where VAPP.EXE is installed. This makes it easy to share some files between all experiments (like .FNT files) while keeping others specific to an experiment (like the stimulus files themselves, or the bitmap images).

4. OPERATION

Once invoked, VAPP reads the stim file and parses it into tokens, compiling and translating these into an internal, memory resident format. Next, the VGA screen is blanked to the solid background colour or, if the **sbg** option is in use, the background image is drawn and displayed. At this point, VAPP is ready to begin stimulus presentation.

If the `waitforkey` option is enabled, VAPP waits with the background image in place (if any) until a keyboard key is pressed, otherwise it launches immediately

into stimulus presentation as soon as it is ready. During this phase, only the Escape key alters program execution, and only if the `escapequits` option is enabled. any other keystrokes are ignored (unless `setinput` includes the keyboard, in which case they will cause log file entries and serial port output when `echo` is enabled) . As each stimulus is drawn, a synopsis of the stimulus is written in the log files.

After the last stimulus has been presented, the program clears the last image from the screen and returns to the DOS command prompt.

Here follows a more detailed description of various aspects of the operation of VAPP.

4.1 Codes

The codes associated with stimuli in the stim file are output via the standard serial port on the PC, chosen with the “setoutport” configuration option. A single byte is output at the onset of presentation of each stimulus, whose value is specified on the stim file line for that stimulus right after the presentation duration (see section **8.2.3, Code**). This limits the codes that can be output to 0 through 255 in the current implementation. For the receiving device to record accurately the time of receipt of the byte, the UART on the receiving device should be programmed to generate an interrupt for each incoming character, and the serial port interrupt handler modified to record the clock time to sufficiently high resolution. Communication parameters should be set to 9600 baud, no parity, 8 data bits, 1 stop bit.

Codes are emitted when the first raster line in the frame containing the stimulus image is output, to within 0.5 ms. This will be immediately after the vertical blanking period.

The manner in which the video system operates imposes a temporal quantization on the SOAs and hence code intervals. This “frame quantization” also affects the duration of presentation of stimuli, since stimuli must appear for an integral number of frames. The situation is exacerbated by the fact that the phosphors used in the video monitor have an exponential decay characteristic which can take many tens of milliseconds to decay below the threshold of detectability. Hence, the use of long-persistence monitors is not recommended.

4.2 Errors

Errors that occur during the execution of VAPP are divided into two classes; those during the reading and compilation of the stim file, and those during the presentation phase (execution of the memory resident stim and image information). The former are termed stim compilation errors, while the latter are termed execution errors. All error messages are written to the log file with the line number of the offending line in the stim or pgi file to aid with debugging.

4.2.1 Compilation Errors

Substantial checking is performed during the compilation process to ensure the correctness of the commands in the stim file. When there is a syntax error in the

stimfile, the logfile will include an error description with the exact line and column in the stimfile where the error was detected.

Note that the file names for font, image, sound, and pgi files are *not* validated until the run is under way, so you should always pre-run your stim files before trying to gather data with them.

4.2.2 Execution Errors

While many errors can be detected during compilation, others can arise during presentation when the memory-resident commands are executed. Timing errors will occur if drawing of images takes longer than the SOA, while certain graphics operations can also result in errors. Execution errors are also fatal and cause termination of VAPP.

5. VGA GRAPHICS ENVIRONMENT

A generic SVGA adapter in the PC/AT is used to generate the video images that are displayed on the colour monitor.

5.1 Resolution and Coordinates

The display screen is divided into 480 raster lines of 640 horizontal pixels each. Since the electron beam scans from top to bottom, the native screen coordinate system ranges from 0 to 639 in the x direction (going from left to right), and from 0 to 479 in the y direction, y increasing downward on the screen. These screen coordinate units are used when specifying screen positions or extents. Graphics routines implement clipping to the screen boundaries and drawing beyond the boundaries can be attempted without error (or effect).

The VAPP graphics routines (as well as the related pgi file graphic commands, described below) employ the notion of a current coordinate. This can be thought of as an invSOAble cursor at a certain x-y screen position. Many pgi commands use the current coordinate as a reference point for the operation they perform. For example, the **lineto** command draws a line from the current coordinate to the coordinate specified in the arguments to the **lineto** command, with the endpoint becoming the current coordinate for the subsequent command(s). The current coordinate is also important for the stim file options discussed under section 8, **Stim Files**, below.

In addition, coordinates or pixel extents are sometimes interpreted as relative to the current coordinate, in which case the values are added to the current coordinate to arrive at the actual screen location. For example, the xoff and yoff stim options are relative to the center of the screen, and if in the above example the pgi command had been **rlineto**, the line would have been drawn from the current coordinate to the current coordinate plus the argument values for the **rlineto** command.

5.2 Colours and Palettes

Present video displays handle colours in one of two ways: by specifying RGB values directly for each pixel, or by using a look-up table called a palette. VAPP 1.0 exclusively uses the latter scheme, where the colour value you specify for a given pixel is interpreted as an index in the palette where the actual RGB values are found.

All the pixels for an entire screen are stored in a video frame buffer, and each pixel can take on values from 0 to 255 (inclusive). The value at each pixel, termed the pixel value or colour value, is used to index into a table of red, green, and blue values which are used when the pixel is displayed. Since there are 256 possible values for each pixel, 256 colours can be displayed on the screen at a time. However, these 256 can be any of 262,000 colours when the VGA adapter is used. For more information on palettes, see Appendix 1: Default VGA Colour-mapping Palette.

When one specifies a colour in a VAPP stim file or in a pgi file, it represents the pixel value that will be employed. Hence, one must also know what palette is in use to unambiguously determine the colour that will be displayed. Presently, only the default palette is supported when drawing with graphics commands.

5.2.1 *Bitmaps, background images, and palettes*

There is an additional concern when loading bitmap images, either in the background (with the sbg config option), or with an image= statement in a stim file. Many .BMP files contain a custom palette which must be loaded into the VGA display palette at display time for the image colours to look correct. This is typical of 256 color bitmaps. VAPP supports custom palette loading with BMP files, but the palette that is loaded will displace the standard VGA palette during the display of that frame. Text and other drawing elements overlaid on top of a BMP with a custom palette will not appear in the same colours that they would have if the standard palette were loaded. The only way to tell for sure what you will get in these cases is to pre-screen the stimfile before use. If the custom palette that loads with a given bitmap prevents you from getting the colours you want in overlaid text and drawing commands, you will have to specify the colours of those drawing commands using entries from the custom palette.

There is an additional concern when loading a bitmap into the background with the "sbg" option. When the background image is loaded at startup time, there is no way to tell what the palette will be for a given stim during the run. For text and line-drawn images, the standard palette will be in place, for bitmap images with custom palettes, the custom palette will be in place. When using the sbg option, one of two things can happen when the sbg image is copied to the background of the display page before other elements are rendered on top. Either the background image's colors can be mapped into the standard palette on a "closest match" basis, or the background image can be displayed with untranslated pixel values and just wind up with whatever colors that happens to generate with the current palette. The choice of which of these two things is done is controlled with the "textpalette" option. If textpalette=0 (the default), then the sbg image will use whatever palette belongs with each stim (which will be the standard palette for text & drawing stim, or the custom palette loaded with a 256

color bitmap). No translation of color values takes place in this case, and you can get strange looking colors in your background image, especially if you are doing text stimuli. The advantage of this is that if your background image and your stim images have the same palette and you are doing mostly bitmapped images for your stim, the background image will have the correct colors.

If `textpalette=1`, the background image's pixel values will be re-mapped to the closest color match values that can be found in the standard palette. This will give reasonable looking background images behind text and line drawn stimuli, at the price of giving strange colors behind bitmap stim that have custom palettes.

To help understand what images are going to look like, I have included a simple utility to extract and display the numerical values in the custom palette associated with any .BMP format bitmap file. In your VAPP directory, there should be a program called PALETTE.EXE, which you call from the command line with the name of the bitmap file you want to examine.

6. SOUND PLAYBACK ENVIRONMENT

VAPP is capable of playback on any 100% Sound Blaster compatible sound card. Any DOS drivers that come with your card should have been installed before you use VAPP. You can check if this has been done in DOS by using the SET command and checking if an environment variable called "BLASTER" has been set. A typical value might look like:

```
BLASTER=A220 I5 D1 H5 P330 T6
```

Though the actual values will vary from card to card. The DOS install program for the Sound Blaster will be provided with your VAPP diskettes, but if it is missing, it can be downloaded free of charge at any time from:

<http://www.download.com/pc/software/0,332,0-40890-s,1000.html?st.dl.subcat82.list.tdtl>

This program must be run from the DOS environment where you will be running VAPP for it to correctly detect your card's settings. It will NOT work correctly if you run it within Windows 95+ and then dual boot back to DOS.

6.1 The WAV format

The VAPP program can only play sounds that are in the Windows standard file format. These files are identified by the extension "WAV". File size is presently only limited by the available memory.

6.2 Mono and Stereo files

Both mono and stereo files are supported, at standard sampling rates up to 44kHz.

7. DISPLAY, TIMING, AND BUFFERING ALGORITHMS

This section describes the algorithms used to draw stimuli, buffer the images for display, and time the SOAs and presentation durations.

7.1 Frame Buffering

Images are drawn in special memory areas called video frame buffers. The term “frame buffer” derives from the notion that the memory area contains all the pixel information needed to display an entire frame of video information. A video frame is one complete vertical scan on the video monitor; and the VGA default frame rate is 60 Hz. Note that this is the origin of the “frame quantization” mentioned in the “**Codes**” subsection under “**Operation**”, above. The frame buffers are special areas of memory physically located on the VGA adapter, and enjoy fast access, inter buffer copying, and ability to be displayed as a set of pixels on the VGA monitor.

When operated at a resolution of 640 by 480 with 256 colours, a standard SVGA card supports 12 video frame buffers when 4 Mb of memory is installed. VAPP always uses one of the frame buffers to store the background image, even if it is just a solid colour. As a result up to 11 stimulus images can be “on board”, that is, drawn in the frame buffers prior to their time of display.

The number of available frame buffers that can be used for predrawing stimuli affects the rate at which complex stimuli can be presented. The basic algorithm that is used by VAPP is to pre-draw as many stimuli as possible in the frame buffers, and then switch the display to the appropriate frame buffer when the starting frame for that stimulus arrives. After that stimulus has been displayed for the specified number of frames, the frame buffer for the image is freed so that another stimulus can be prepared. Hence, it is possible to rapidly present complex images as long as enough frame buffers are available to allow one to trade off the SOA with the drawing time for the images.

7.2 Timing Considerations

As briefly mentioned when event codes were discussed (above), the nature of the video system imposes a “frame quantization”, on all timing parameters. A “frame” is one vertical scan period of the display, and usually takes 16.666 msec. The vast majority of the 16.666 msec elapses while the electron beam scans from the top of the screen to the bottom. However, remaining time in each cycle is used to reset the beam to the top of the screen in preparation for the next cycle. This period is termed the vertical blanking period, since the beam is shut off, or blanked, during this period. This is the period in which the frame buffer that is to be displayed during the next scan is selected.

One is often interested in determining the minimum SOAs and/or durations that can be employed when presenting stimuli. These parameters are determined by the drawing routines and the combinations of options, and are directly related to the time required to draw the stimuli. In general, the number of separate images in a stimulus and their complexity determine the time that is required to draw an image. Usually one empirically determines whether a particular set of stimuli in a given stim can be presented simply by trying it.

8. STIM FILES

Stim files specify the images and sounds that constitute stimuli as well as the corresponding codes, SOAs, and presentation durations. They are standard text files and can be generated with any text editor. Usually however, one uses the **SG** program to automate the generation of stim files. The resulting stim files can be altered if needed using a text editor. For simple stimuli, such as text, all information needed to run VAPP can be contained in the stim file. For other applications, one may need to generate images in separate files which are referred to in the stim file. During the compilation phase all of these files will be read and placed in memory for use during the presentation phase.

Supplementary images can be “parameterized” as sequences of lines, points, polygons, etc., described in a text file. These files have a specific format as described below and are termed pgi files for parameterized graphic image.

It is also possible to employ files that contain a description of the actual sets of pixels to place in the frame buffer for the stimulus display. These are raster images, and are kept in a binary file called a bitmap image. The .BMP extension indicates a standard format for bitmap images that is readable by Windows Paintbrush and nearly all other graphics programs. Another related graphics standards readable by VAPP include PCX and JPG, both identifiable by their file extension.

One can modify an image on a line in the stim file by using certain options. Some options enable one to employ the same basic image but present it at different locations and/or different colours. The remainder of this section describes the syntax and semantics of the stim file in detail.

8.1 General

Stim files are line-oriented in the sense that all the basic information about a stimulus is contained on one line. The order of presentation of stimuli is the same as their sequence of description on lines in the stim file. Since lines can become uncomfortably long when describing all the information about a stimulus, you may indicate that several lines in the text file are to be treated as a single “line” for purposes of parsing. This is useful if you wish to specify multiple images or sounds that should be presented as a single stimulus. A stimulus then is described by a basic line and one or more continuation lines. A line is continued if the last valid option on the line is a plus sign, ‘+’.

Within each stim line, there is a necessary grouping of commands by device. All graphics commands must precede all of the sound commands. Placing commands out of order will generate an error message.

Each line of input is parsed into tokens, which are sequences of contiguous non-white characters separated by spaces or tabs. A stimulus has four mandatory arguments on the basic line: an integer SOA, a presentation duration, an associated code, and the basic stimulus. The arguments must appear in this order as the first four tokens on any basic line. Additional arguments on the line prior to any comments are options that in some way modify the drawing of the stimulus or its display. Continuation lines have a similar structure, except that

the SOA, duration, and code arguments are not needed and should not be present; hence continuation lines begin with an image and the remainder of the functional arguments represent options.

To improve the readability of stim files one can employ comments, indentation, and additional spacing between arguments if desired; this does not affect the parsing of arguments and is strongly encouraged.

8.1.1 String arguments

A character string required for a command argument must be surrounded by double quotes if it contains spaces. There is no way to put a double-quote character in a string, it is always treated as a string delimiter. You may put carriage returns in a quoted string by putting a carriage return in the stim file between the opening and closing quotes.

8.1.2 Comments

The stim file can contain comments - a semicolon anywhere in a line will make the rest of that line a comment. Also, 'C' style comments are allowed anywhere - the combination `/*` begins a comment and `*/` ends one. Comments of this kind may span several lines.

8.2 Mandatory Arguments

Each basic line for a stimulus must have the four mandatory arguments to describe the time interval between onset of the current and the next stimulus, the duration of presentation, the code to emit, and the basic stimulus itself beginning the line. Additional details regarding each of these mandatory arguments are detailed in the next four subsections.

8.2.1 SOA

The first argument on a basic line is the interval from the beginning of the display of the stimulus to the beginning of the display of the next stimulus. This interval can be specified as either a positive integer representing a number of milliseconds, or the letter 'f' or 'F' immediately followed (no space) by a positive integral number of frames. Due to the "frame quantization" effect (described in more detail in the section **4.1, Codes**, above) SOAs specified as a number of milliseconds will be rounded up to an integral number of frames. If, for any reason, the number of frames would be less than one, it is forced to one with a warning during the parsing of the stim file.

Note that the timing of the SOA is from the end of the *actual* presentation of the stimulus, not the requested duration. This means that the roundoff errors in the durations and SOAs are cumulative. If you have a frame time of 16 ms, and requested display of an image for 24 ms with an SOA of 24 ms, each of these would be rounded up to 2 frame times, or 32 ms each. The next stimulus would therefore start after 64 ms, not the 48 that would be expected without time quantization imposed by the periodic nature of the video display. If this same stimulus were repeated, say, 100 times, then the duration of the stimulus run

would be 1.6 seconds longer than the 4.8 seconds expected. Unexpected results of this kind can be avoided by using the F1 notation for stimulus durations in terms of frame times. Knowing the durations of stimuli then depends on accurately knowing the frame period (scan time *plus* vertical retrace interval).

It should be noted that the interval described here (SOA) is more commonly and more correctly referred to as the SOA (Stimulus Onset Asynchrony). Calling this interval the SOA is a convention inherited from the VVSP.

8.2.2 Presentation Duration

The second argument on a basic line is the duration of presentation of the stimulus in milliseconds. It too is quantized in units of frames and can be specified as a number of milliseconds (which is rounded up and converted to a number of frames), or the number of frames directly, by immediately preceding the positive integer with the character 'f' or 'F'. If, for any reason, the duration of presentation would be less than one frame, it is forced to one frame. Additionally, if it would exceed the associated SOA, it is forced to equal the SOA. In both cases, a warning is issued during compilation.

When a stimulus comprises both a sound and an image, the triggering of the sound is delayed if necessary until the image is presented. The sound is triggered at the same time that the code is transmitted out the serial port. If a sound is the only stimulus on the stim file line, then it is not subject to the time quantization imposed by the video hardware.

For a further discussion of sound trigger timing, see section **8.2.3, Code**, below.

8.2.3 Code

The third argument on a basic line of a stim entry is the code that will be emitted when the stimulus is presented. Any positive integer is allowed on the range 0 to 255 inclusive. For an image of any kind, the code is output at the serial port when the electron beam in the monitor is on the first (topmost) scan line in the image.

8.2.4 Basic Stimulus

The fourth argument on a line is the basic stimulus, and the same syntax is used to specify additional stimuli on continuation lines. A stimulus specifies one of a number of major classes of sounds or images as well as the specific data file itself. Currently, the classes of stimulus are text, image, pgi, and sound, and each stimulus argument has the form:

```
class=string
```

where class is one of text, image, sound, or pgi. Note that data files are searched for first in the current directory, then in named subdirectories under the current directory, then under the VAPP installation directory. A further description of file locations is available under section **3.3, File locations**.

The class keyword (e.g. text, image, pgi, sound) is case sensitive and must be entered in lower case.

Here are a few examples of syntactically valid stimulus image specifications:

```
text=coffee
text="Hello World, here I come!"
image=monalisa.bmp
pgi=pentagon.pgi
sound=hello.wav
```

Note the use of the quoted string in the second example to enable the inclusion of spaces in the text string.

The sound command must follow any visual stimulus commands on the command line.

8.3 Options

Arguments following the stimulus image specification on a line in the stim file specify options that modify the basic stimulus or some aspect of its presentation. These options allow one to employ the same image while, for example, drawing it at different locations on the screen, or in a different colour. Note that when an argument is encountered that begins with a ';' (outside of a quoted string), the rest of the line is considered a comment. Hence, option arguments must precede any comments. Except for the continuation option, the order of appearance of the option arguments is irrelevant, as long as they follow the stimulus image specification and precede comments (if any) on the line. The option keywords (e.g. xoff, yoff, esp, colour, lblo) are case sensitive.

The stim file options alter a parameter or action that will otherwise have a default value; as set either in the .CFG file, or on the command line when VAPP was invoked. Manipulating options in the stim file follows the same syntax and rules as used in the CFG file, with the exception that the options are all specified on a single line. Continuation lines should be used for long strings of options to maintain readability.. One may also employ pgi files, which can alter the entire plotting environment.

Note that options specified on the stimulus line in the stim file are only in effect for the duration of that stimulus. When the stimulus duration elapses, the current default is restored.

Here is a list of some of the salient parameters that can be preset using stim file options and their associated default values.

Parameter	Default Value
Current Coordinate	xoff=0, yoff=0 (upper left)
Foreground colour	15 (white)
Background colour	0 (black)
Font	Roman.16.14 (or option)
Label Origin	5 (Centered, vertically and horizontally)
Palette	default (or option)
Drawing Mode	DMSET

Parallel output	outpp=0 (or option)
Audio volume	volume=0..255
Audio balance	ear= r b

Options are most useful for basic image types of text or image, where re-positioning the image or selecting a colour is all that is usually required to draw the desired image. Pgi image types entail a file of graphics commands that can be written so that they either do or do not override options from the stim file. The specific experimental requirements must be considered when deciding whether a stim file option should control a parameter or whether it should be selected in the pgi file itself.

Each stim file option and its syntax is described below.

8.3.1 Xoff and Yoff

These two options alter the location on the screen where the basic image will be drawn, assuming the image specification allows this possibility. The xoff and yoff keywords specify an x offset and y offset respectively from the center of the screen that will become the starting location for drawing the stimulus image. None, one or both of these positioning options can appear in any order, and have the form:

```
xoff=nnnn
```

where nnnn is an integer that will be added to the corresponding coordinates for the center of the screen prior to drawing the stimulus image. Note that xoff and yoff options imply a relative move, and hence nnnn can be positive or negative. The value should be in units of screen coordinates, and should not result in the current coordinate being off the screen.

8.3.2 Colour

One can alter the current foreground colour that is selected prior to drawing the stimulus image by including a line of the form:

```
colour=nn
```

where nn is an integer between 0 and 255 (inclusive). These colour values are used to index into the mapping palette to derive the actual colour that is displayed, so this option interacts with the palette that is currently in use. Appendix 1 lists the colour values and the associated display colour for the default VGA palette.

8.3.3 Label Origin

The label origin, which determines the position of text relative to the current coordinate, can be modified by including the option:

```
lbo=nn
```

where nn is an integer specifying the new label origin (see Appendix 2 for a description of the various label origins).

8.3.4 Font

While one can specify a default font for **text** or **pgi label** commands, it is also possible to dynamically and temporarily select a different font which will be used to draw text. The current font, like other options, is reset to its default value prior to drawing each stimulus image. One selects a font by an argument of the form:

```
font=filename
```

where filename is the MS-DOS style file name for the desired raster font.

Fonts are not stored in memory during the compilation phase of VAPP in the current implementation. Hence, the inability to open the specified font file during stimulus presentation or the specified font file having the wrong format results in an execution error.

8.3.5 Parallel output

The **outpp** option specifies a value to be sent out the parallel port (LPT1) at the onset of the stimulus. The value is specified as a string representation of a number on the range 0 to 255 so for example use

```
outpp=122
```

to send hex value 7A out the parallel port (bit pattern 01111010).

DB25 pinout for parallel port:

1	~Data strobe
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	~Ack
11	Busy
12	Paper Empty
13	Select
14	Afd
15	~Error
16	~Reset
17	Select in
18-25	Ground

These pin numbers can usually be read directly from the molded plastic on the cable end, or on the panel mount socket at the back of your computer, though you often need a bright light and a magnifier to read them.

A name with a tilde (~) indicates inverted logic. The byte indicated in the outpp command is placed on the Data 0 through 7 lines. The logic is noninverted, meaning that if you program a bit to 1, the voltage on that pin is 5V, logic 0 gives

0V. The least significant bit appears on D0, msb on D7. The other lines are controlled and read through the parallel port status and control registers. They are concerned with printer control, and are not programmed or read by VAPP.

8.3.6 Volume

The sounds played back on the Sound Blaster have rudimentary volume control. There are 16 steps of volume divided among the range 0..255, so any value of the volume between 0 and 15 will give you the quietest playback, and anything between 240 and 255 will give you the loudest. Note that the volume setting remains in effect until the next time it is set, so if you want to play a single sound at half volume, you would set "volume=128" on that sound, and then "volume=255" on the following sound. Also note that the volume and balance commands must precede (be to the left of) the sound that they apply to on the command line.

8.3.7 Balance

The **ear** command provides simple control over which ear will be presented with the sound being played back. It can be set to **l** for the left ear, **r** for the right ear, or **b** for both. You may use upper or lower case. As with the volume command, it stays in effect until set again, so you must set it back to **b** to restore normal playback. The **ear** command works equally well for both mono and stereo WAV files.

8.3.8 Continuation

To enable one to specify multiple component stimuli as constituting a stimulus, the '+' (plus sign) alone as the last option argument on a line in a stim file indicates that an additional image's specification and options(s) continue on the next line in the stim file. Note that the '+' must be the last valid option argument (prior to any possible comments). A stimulus can entail as many stimulus images as necessary by using additional continuation lines.

A continuation line, unlike a basic line, does not include SOA, duration, or code arguments. Hence, continuation lines have the same structure as a basic line starting at the stimulus image specification.

Note that there is not a one-to-one correspondence between graphic entities that are included in a stimulus and the number of lines in the stim file that specify that stimulus, since **pgi** or **image** files can specify any number of distinct images.

Images specified on continuation lines in the stim file are drawn in the order that they are listed, so for areas where they overlap, the last one drawn will be vSOAble.

8.4 Stimulus Types

A stimulus specification currently can be **text**, **image**, **pgi**, or **sound**, selecting a text string, a bitmapped image, a parameterized graphics image file, or a sound file as the type of stimulus (described above). Each of these has different

characteristics that require different treatments during presentation as well as different methods of image preparation.

8.4.1 Text

The text type of stimulus image is the simplest; the associated string itself is used as text that is displayed. It is displayed using the default parameters (i.e. middle of the screen, in white) as modified by any options on the line in the stim file.

8.4.2 Image

Image files contain an encoded representation of a rectangular region of pixels. The the only file types currently supported are .BMP, .PCX and .JPG, and a file for display must include one of these extensions on the file name. When other file types are supported in the future, the file extension will be used to indicate the file type. A number of programs are available to convert images from other software in other forms to BMP or PCX format for use with VAPP.

Generally, bitmap files can be decoded and drawn on the screen at any position. When reconstructed and drawn on the VGA screen by VAPP, the center of the rectangular region corresponds to the current coordinate. Hence, without any xoff or yoff options, they will be placed in the center of the screen.

8.4.3 Pgi

Pgi (parameterized graphics image) files form a means of specifying images by primitive operations that are executed by VAPP to draw the image. Pgi files are most useful for representing low information content images that can be easily parameterized (e.g. in terms of lines, polygons, rectangles, and text), and have modest memory requirements.

Pgi files inherit all the stim line options when they are executed, and the pgi file can employ this information or not, to achieve the desired result. For detailed information on writing pgi files, see section **9 Pgi Files** , below

8.4.4 Sound

To play an audio clip, include the 'sound' command after any visual commands on a stim line. You must use the whole name of the file, including the .WAV extension. Note that any **volume** or **ear** options should precede the sound command.

Sound files are mono or stereo sampled pre-recorded sounds available on the VAPP computer's hard drive for playback. Sound files are searched for first in the current directory, and then in the SOUNDS subdirectory under the current directory. The only file format currently supported is the Windows WAV format, and the only card supported is the Sound Blaster by Creative Labs. For a discussion of file formats and limitations, see "Sound playback environment" on p. 15.

9. PGI FILES

Pgi (parameterized graphic image) files are text files containing sequences of graphics commands. They can be generated with any text editor, and can encode arbitrarily complex images, given sufficient image design and preparation. To effectively author pgi files, one should be familiar with section 5, **VGA Graphics Environment**, above, as well as this section.

9.1 General

Pgi files are line oriented and are similar to stim files in the way commands and arguments are parsed on each line of input. Briefly, blank lines are ignored, and tokens on a line are separated by spaces, or tabs.

VAPP compiles pgi files into an internal sequence of VGA graphics primitives. During the presentation phase, these commands are executed in the same sequence as they appeared in the pgi file.

At any point in a stimfile, you can include other command files by using one of the tags "include", "pgi" or "subimage" followed optionally by an = sign, and then a file name. Commands from this file are processed as part of the current image up to the end of the included file, after which processing continues at the next token on the original stimfile line. Included files may include other files to any depth, though there is no check for circular includes.

9.2 Pgi Plotting Environment

The execution of pgi commands takes place in an environment of globally available state information that controls the colour, line type, fill pattern, etc. used by drawing commands. The environment consists of parameters in the following table, along with their initial value.

Parameter	Default Value
Current Coordinate	x = 319, y = 239
Foreground Colour	WHITE (1)
Background Colour	BLACK (0)
Label Origin	Center (5)
Drawing Mode	SET (0)
Font	Roman.16.14 (Inv. Option)
Palette	Appendix 1 (Inv. Option)
Line Type	Solid (0xFFFF)
Polygon Fill Pattern	Solid (0)
In Polygon Definition	No

Those parameters marked (Inv. Option) can have their default values selected by invocation options on the VAPP command line or in the CFG file.

Before a pgi file that is specified in a line of a stim file is executed, the plotting environment is set as indicated in the above table. Then, any associated options in the stim file are executed, and can alter the font, foreground colour, or current

location. Then, execution of the pgi file begins. If, during the execution of a pgi file another pgi file is invoked (using the **subimage** command), the entire plotting environment is copied into a new area and becomes the plotting environment for the execution of the nested pgi commands. Hence, nested pgi files inherit the plotting environment from their callers. However, when execution of the nested pgi file is complete, the new environment is discarded and the caller retains the state of its plotting environment before the call. Thus, a type of last-in first-out stack of plotting environments is maintained by VAPP during the execution of PGI files.

The one state variable that is not kept in the stacked plotting environments and is thus “global” between nested executions of pgi files is the polygon definition flag, and the associated polygon definition. Since polygon processing requires substantial memory, only one polygon can be defined and held at a time. So, if one pgi file defines a polygon and then calls another pgi file that defines another polygon, when execution resumes in the calling pgi file the original polygon definition will have been replaced with the called file’s definition. This is not necessarily a disadvantage, since it allows one to define polygons in subimage pgi files, and invoke them to define a polygon which can be subsequently filled or outlined.

9.3 Commands

The PGI files actually permit any of the stim commands (including the pgi command itself), though they are typically used to encapsulate complex sequences of drawing commands for easy re-use. As with drawing commands that are issued from within the stim file itself, any options that modify the drawing environment should be issued *before* the drawing commands that they affect. For example, you should issue a font command *before* (to the left of) the text command it applies to.

Some commands (e.g. **lineto** and **moveto**) have relative forms, in which the argument values are added to the current coordinate in order to arrive at the actual destination. For these commands, the relative form has the same name as the absolute form with an ‘r’ prefixed to the name. In the following descriptions, the command name is set in boldface.

9.3.1 State Commands

A number of commands set a parameter in the current plotting environment rather than actually cause any drawing on the screen. These commands take a single integer argument, and do not change the current coordinate. Different drawing commands employ different combinations of these parameters and their current values at the time they are executed. See section **9.2, Pgi Plotting Environment**, above, for more details.

9.3.1.1 **setfgcolour** pixelvalue

The **setfgcolour** commands sets the current foreground colour to pixelvalue, a number between 0 and 15, inclusive. Many commands use the foreground colour as the value to which pixels are set during execution. The pixel values are

used as indices into the current palette to obtain the actual colour that is displayed on the screen. Appendix 1 has a list of the default palette mapping.

9.3.1.2 **setbgcolour** pixelvalue

The **setbgcolour** is similar to the **setfgcolour** command, except that the current background colour is set to pixelvalue.

9.3.1.3 **setlinetype** mask

The integer mask is used as a 16 bit mask which is used by the **lineto** command to create dotted, dashed, and other types of lines. The line pattern is used to determine which pixels in the line get drawn depending on which bits in the pattern are set. The pattern is a 16-bit value, and everywhere that a bit is set to a 1 in the pattern, a pixel will be drawn in the line. Everywhere that a bit is a 0, the pixel will be skipped in the line. Note that bit 0 in the stipple pattern corresponds to pixel 0,16,32,... in the line, bit 1 is pixel 1,17,33 etc. To create a line that is drawn as a 'dot dot dash dash' you would use the following value:

0011100111001001b or 0x39C9

For dotted lines, use -21846, for dashed, 3855, and for solid lines, use -1.

Note that patterned lines can only be 1 pixel wide, and the pen size will be ignored when drawing a stippled line.

9.3.1.4 **setdrawmode** drawmode

The drawing mode is a state variable that indicates the way a pixel value should alter current values in the frame buffer when drawing occurs for certain commands. The valid values for drawmode are

- | | |
|----|---|
| 0 | set the frame buffer to the pixel value, |
| 8 | logically 'and' the pixel value with the current value in the frame buffer and place the result in the frame buffer |
| 16 | logically 'or' the pixel value with the current value in the frame buffer and place the result in the frame buffer, |
| 24 | logically 'exclusive or' the pixel value with the current value in the frame buffer and place the result in the frame buffer. |

Note that the exclusive or function is invertible; repeating a drawing operation on the frame when the drawing mode is 24 restores the pixel values to the state they had before the first drawing operation.

9.3.1.5 **selectfp** fillpatternindex

One selects the fill pattern for the **fillpgon** command using **selectfp**, where fillpatternindex is a small integer corresponding to the number of the desired fill pattern. The currently available fill pattern indices are:

- 0 - solid
- 1 - 50% checker

- 2 - Large diagonal criss-cross
- 3 - diamond shaped blobs
- 4 - Large vertical criss-cross
- 5 - 10% checker
- 6 - Small vertical criss-cross
- 7 - Round blobs

A number out of range is brought into range by taking the remainder of the supplied argument after dividing it by the number of fill patterns.

9.3.1.6 **setlblorig** labelorigin

The current label origin is selected using this command, here labelorigin is one of the label origin reference locations that are specified in Appendix 2. The label origin is used only by the **label** command, and controls the relative location of text that is drawn with respect to the current coordinate.

9.3.2 **lineto** *xc yc*

9.3.3 **rlineto** *xc yc*

The **lineto** commands draw a line from the current coordinate to the xc and yc coordinate location using the current foreground colour, line type, and drawing mode. The end point of the line becomes the current coordinate.

9.3.4 **moveto** *xc yc*

9.3.5 **rmoveto** *xc yc*

The **moveto** commands only move the current coordinate to the location specified in xc and yc.

9.3.6 **orect** *nxpix nypix*

The **orect** draws the outline of a rectangle with a width of nxpix and a height of nypix, with the current coordinate located at the upper left corner of the screen. Both nxpix and nypix should be positive integers, and the coordinates of all four corners of the rectangle should be on the screen. The current coordinate is unchanged.

9.3.7 **frect** *nxpix nypix*

The **frect** is similar to the **orect** command, except that the entire rectangular area is filled with the current foreground colour. Note that only solid fills can be obtained with this command; rectangular regions with arbitrary fills should be implemented using the polygon commands. The current coordinate is unchanged.

9.3.8 *image bmpfile*

One can read an image file and reconstruct it on the screen using this command. Bmpfile is the name of the file containing the image. If it is a monochrome image, the current foreground colour is used to set displayed pixels; colour image files contain the pixels values for all points on the screen. See section **8.4.2 Image** for more information. If the specified file cannot be opened or is not a valid image file, in execution error is generated. The current coordinate is not altered by the execution of a **dispimg** command.

9.3.9 *label "text string"*

The **label** command places the argument string on the screen using the current font, foreground colour, and label origin. It is a synonym for the "text" command in the stim file. The current coordinate becomes the reference point for the label origin. Note that double quotes (") can be used to represent strings containing separators. The current coordinate is not affected by the label command.

9.3.10 *font fontfile*

The named font (fontfile) becomes the current font for all subsequent **label** commands if the **font** command is successful. An execution error will occur if the named file is not a valid font or does not exist. The current coordinate is not changed by the **font** command.

9.3.11 *pgi pgifile*

One can effectively employ subroutines in pgi files by re-using the pgi command. The pgifile is opened and used as the source of commands until the end of the file is reached, at which time reading of the current pgi file continues. The called pgi file inherits the current plotting environment of the caller, including font, foreground colour, fill pattern, current coordinate, etc. (see section **9.2, Pgi Plotting Environment**), but the caller's environment remains unaltered across the call; thus, the current coordinate remains the same.

Pgi files can be nested to any level desired as long as sufficient memory space is available. The use of subimages can greatly reduce the memory requirements for stimuli, especially when images are constructed using many common elements.

If the named file cannot be read or an error occurs during its execution, an execution error occurs in the caller.

9.3.12 *Polygon Commands*

The VGA library and VAPP provide a simple polygon system that can be used in pgi files. Only one polygon can be defined at a time, and definitions are not stacked in the plotting environment (see section **9.2, Pgi Plotting Environment**). However, a polygon can consist of up to 64 subpolygons with a total of 1024 vertices in all. A polygon must first be defined using **startpgon**, **lineto**, **moveto**, and finally **endpgon** commands. It can then be filled with a fill

pattern, or outlined using the current line type, using the current foreground colour. Here follows a detailed description of the polygon related commands:

9.3.12.1 Startpgon and Endpgon

The start polygon definition command does not require any arguments, and places the VGA plotting routines in the “polygon in definition” state. When the **startpgon** command is executed, the current coordinate becomes the first vertex, and subsequent **lineto** commands do not cause any drawing, but rather add the coordinates as another vertex for the sub-polygon. It is termed a sub-polygon, because if a **moveto** is encountered in the sequence of **lineto** commands, the current sub-polygon is closed, and a new sub-polygon definition begun, using the coordinates of the **moveto** command as the first vertex of the new sub-polygon. A sub-polygon must be closed, and this is enforced by assuming the last coordinate in each definition joins with the first. Polygon definition ends when the **endpgon** command is executed, thus taking VAPP and the VGA plotting routines out of the “polygon in definition” state, with the single defined polygon composed of all the sub-polygons that were defined. After the **endpgon** command, the current coordinate is as it was before the **startpgon** command.

The coordinates used in the **lineto** and **moveto** commands that are encountered while a polygon definition is in progress need not be valid screen coordinates, since the vertices will be translated by the arguments to subsequent **outlinepgon** or **fillpgon** commands. When drawn, however, all the vertices of the polygon should lie on the screen. The coordinates in a polygon definition thus form a sort of relative displacement from the reference point supplied when the polygon is drawn.

Note that if a sub-polygon definition consists of less than 3 vertices, the entire sub-polygon is rejected, and it is permissible to define polygons consisting of only one sub-polygon. Remember, polygon definitions and status are not stacked with the plotting environment.

9.3.12.2 outlinepgon rx ry

9.3.12.3 routlinepgon rx ry

Once a polygon has been defined, it can be outlined using the current line type and foreground colour using the **outlinepgon** command. The arguments rx and ry form a reference location whose coordinates are added to those of each vertex in the polygon before the outline is drawn. In this way, one can draw the same polygon at different locations on the screen without repeating the definition every time. If the relative form of the command is invoked, the reference location is formed by adding rx and ry to the current coordinate, giving additional flexibility.

9.3.12.4 **fillpgon** rx ry

9.3.12.5 **rfillpgon** rx ry

Filling a polygon works much like the **outlinepgon** command, except that the interior of the polygon is filled with the current fill pattern. Filling and outlining a polygon are two separate operations, one can have an outlined but unfilled polygon, a filled but un-outlined polygon, or a filled and outlined polygon.

When filling is performed, overlapping and /or enclosed subpolygons are filled so that areas are alternately filled. For example, if the polygon definition consists of three concentric triangles, the region between the outer and middle triangles will be filled as will the inner triangle, whereas the region between the inner triangle and the middle triangle will remain unfilled.

10. THE LOG FILES

VAPP supports two log file formats, which are created simultaneously. By default, VAPP holds the entire contents of the log files in memory during execution and only writes them out to disk at the end of the stim run, to speed execution. This can be a problem for extremely long runs (several hours or more) on computers with little memory. If you are planning such a run, you should always test it first for "out of memory" errors, as you should pre-run all of you stim runs for correctness.

The two file formats currently supported are called LOG and CSV, after the extensions applied to the files. The LOG file contains a fairly verbose description of the stimulus presentation and subject responses, along with any error messages. It is easy to read and works well for debugging your stim runs. The CSV format is a standard text-only spreadsheet and database format that makes it easy to read the stim performance and response information into Excel or other spreadsheet programs for analysis.

When preparing a new run, always read through the LOG file for any error or diagnostic messages that would indicate that there is a problem with presentation *before* trying to run subjects.

10.1 Name and location

The log files are written in the same location that the stim file was found. The name is a composite of the date and an ordinal count of the session number on the day in question, from 00 to 99. The extensions are always .LOG and .CSV for VAPP Log file. The files are DOS text with hard formatting (spaces instead of tabs). Some sample file names are:

```
96121000.LOG  
97012000.LOG  
97012001.LOG  
97012002.LOG
```

which would have been generated by the sole run on Dec 12, 1996, and the first three runs on January 20th, 1997. If more than 99 runs are done on a single

day, VAPP will detect the presence of the existing log files and exit with an error message on the screen: "Too many log files already present, delete or remove and try again".

You can override the default naming scheme by using the logfile= option on the command line or in the config file. If you name a file that already exists, that file is overwritten without warning. When specifying the name, do not include an extension.

10.2 Contents

10.2.1 LOG

This file contains diagnostic output from the startup and shutdown sequences, as well as a description of the presentation of each stim. During the startup, the version number of the program is put in the file, along with the results of the video timing test. It is in a human-readable text format. It also contains any error messages generated during the run.

10.2.2 CSV

This file has the same base name as the log file, but with the extension '.CSV'. This is a standard delimited file format directly readable by Excel, 123, and other spreadsheet programs. The CSV file only contains information gathered during the stim run, without any diagnostic information about the startup and shutdown. It has a single line of column headers at the top,

```
"Current Time", "Time Occurred", "Code", "Action", "Requested time", "Message/Value"
```

with a column of information beneath each heading. The information in the first column is the time that the entry was made in the log file, NOT when the event took place. The column that you will normally be interested in is the second one "Time Occurred". The Code is the user code you specified for that stim in the stim file. The "Requested Time" is the time that the event was supposed to take place, so you may want to watch for discrepancies between that and the "Time Occurred".

The "Action" is one of the five following events:

- Stim Presented
- Stim Removed
- Key pressed
- Key released
- Button pressed

which are all pretty self-explanatory. The "Message/Value" field will contain the text description of the stimulus for stim presentation and removal, the keystroke for key press/release, and the button number for button presses. If a keystroke is an unprintable character, its ASCII value will be printed after a backslash, eg. \27 for the escape key.

Note that the “Stim Removed” event is only included if you have specified the `csvremove` option in the configuration file or on the command line.

10.2.3 Error messages

The LOG file will contain any error messages from the parsing of the stim file, or any execution errors encountered. Since errors terminate the program, the details of the error will be the last thing in the file. If VAPP is unable to load properly (such as happens when Display Doctor is not installed), then you will get a short log file describing the problem and with instructions for remedying it. If VAPP exits because of an error, the contents of the LOG file will be dumped to the screen, and the error message will be among the last lines.

10.2.4 Stimulus times

During a successful run (and an error aborted run up to the point where the error is detected), the log file will record the actual time of presentation and actual duration of each stimulus, which may vary slightly from the requested SOA and DUR because of time quantization caused by the video hardware. Each line will have the form

```
hh:mm:ss.nnn, nnnn, nn, "stimulus string"
```

where the fields are the start time, the duration in milliseconds, the code specified in the stim file that is output at the serial port, and the basic stimulus line copied verbatim from the stim file. This last field will include the requested SOA and duration, which can be compared to the actual values achieved during the run.

11. SEE ALSO

ADDWORDS	Stim file building utility, used with SG.
SHOWVFPS	Display the available fill patterns and their corresponding indices.
EDITVPAL	Display a VGA palette file and possibly edit it and/or create a new one. The displayed palette can be the default.
SG	Stim file Generator.
VAPPTIME	Computer performance tester which produces the VAPP.CFG file, without which VAPP cannot run. This file contains information needed to customize the performance of VAPP to suit the computer it is running on.

12. POSSIBLE FUTURE ENHANCEMENTS

12.1 Support for other image file formats

The image types supported could be expanded to include, for example, .GIF.

12.2 Motion picture stimulus presentation

The software could be expanded to include support for one or more of the motion picture display types, perhaps mpeg.

12.3 Alteration of default options within the stim file

Options that are available for alteration in the CFG file, and which can be overridden on a stimulus-by-stimulus basis in the stim file, could be added as independent lines in the middle of the stim file. In this way, for example, it would be possible to change the background colour in the middle of a run.

13. APPENDIX 1: DEFAULT VGA COLOUR-MAPPING PALETTE

Colour values specified in the various types of images are not bound to particular colours on the VGA screen, but use a colour mapping table, or palette to determine the actual displayed colour for a given pixel value. The pixel values on the VGA can range from 0 to 255 (inclusive), and thus 256 different colours can be displayed on the screen at a time. The 256 displayed colours can be selected from 262,144 possible colours; this mapping information constitutes a palette.

VGA Default Palette

Pixel Value	Description
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Light gray
8	Dark gray
9	Light blue
10	Light green
11	Light cyan
12	Light red
13	Light magenta
14	Yellow
15	White

14. APPENDIX 2: LABEL ORIGIN POSITIONING SYSTEM

The direction and justification of text is controlled by a three digit number, indicated in the stim file with the “lbo” tag. The three digits independently control one aspect of the orientation each.

1	2	3
Left-right justification: 0 - Left justified 1 - Center justified 2 - Right justified	Top-bottom justification 0 - Top 1 - Center 2 - Baseline 3 - Bottom	Text Direction 0 - Left 1 - Up 2 - Right 3 - Down Text is rotated to match the direction of writing – Left text is upside down, Up text is ‘lying on its back’ and so on.