Basic and Introductory Concepts

Palette

```
typedef struct RGB {
   byte r;
   byte g;
   byte b;
} RGB;
```

Palette entries consist of three bytes - one byte each of value for Red, Green, and Blue. Unlike many PC VGA cards which only use 6 bits of each color vector, the Second Sight supports a full palette range of 16.7M colors; i.e., each R G or B value is 256 levels.

Palette data is required in Second Sight's 256 color modes, as well as in its text modes (which only use the first 16 palette entries). The HighColor and TrueColor modes of the card do not use palettes, because they directly store RGB color information in each pixel.

Image and Pixel Buffers

```
typedef struct LocInfo2 {
      Word portSCB;                    /* $5A5A to indicate a LocInfo2 record */
      union {
          struct {
            Pointer ptrToPixImage;     /* these fields are the same as the */
            Word width;                /* the regular LocInfo record */
            Rect BoundsRect;
          } li1;
          struct {
            Byte flags;                /* 1 = VGA buffer, 0 = GS Mem buffer */
            Pointer ptrToPixImage;     /* offset in VGA memory, or IIGS address */
            Word width;                /* width of pixel map in pixels */
            Byte pixDepth;             /* number of bits per pixel */
            Rect BoundsRect;
          } li2;
      } info;
};
```

The LocInfo2 record is a backwards-compatible extension of the original QuickDraw LocInfo record. It specifies the information necessary to locate the memory address and mask of an arbitrary pixel in a pixel map, either in VGA memory or GS memory. The equation to locate the address of a pixel, assuming a pixel depth of 8, 16, or 24 bits, is:

```
address = (Y * width) + (X * (pixDepth / 8)) + ptrToPixImage;
```

Library Routines

The library routines described below are implemented as a C-accessible library. They should be accessible from any language that can operate with the ORCA™ development system.

The actual implementation of the routines is undefined. Currently it is a simple library, but in the future the core routines may be implemented as an IPC process (which would provide the most flexible and quickest method of accessing these routines such that the code is shared - a necessity in multitasking systems).

```
int vgaStartup(void)
```

Returns -1 if no Second Sight card is found, or if the Second Sight library is not found via IPC.

Returns 0 if the card is found, and the library is properly initialized

`vgaStartup` does a number of things. First, it checks to see if a Second Sight card is installed in the computer. If no card is found, an error is returned.

Second, the routine tries to locate the core Second Sight library; this is generally an INIT file which contains IPC "hooks". If this code is not found, an error is returned.

This call **must** be made before any other calls in this document.

```
int vgaLastError(void)
```

Returns the last error code found by the VGA library. See the section on each library call for a list of possible errors.

```
int vgaCopyPixelMap(int x,int y,LocInfo2 *loc,Rect *mapRect)
```

x,y	x & y coordinate on VGA screen to place pixelmap
loc	pointer to a LocInfo2 record specifying the attributes of the source pixel map.
mapRect	pointer to a Rect structure describing the portion of the source pixelmap to copy.

This function copies a portion of a color pixelmap (the pixel depth is specified in the LocInfo2 record) to the VGA display. No clipping is performed other than insuring that no data is copied outside of the VGA memory specified by the global VGA LocInfoII record.

The pixel depths of the two bitmaps must be the same, and pixels may only be copied on at least a one-byte boundary. See the function vgaPixelMapToPort for a function which will color-expand

```
int vgaSetMasterLocInfo(LocInfo2 *loc)
```

This function sets the master VGA LocInfo record. This record defines the characteristics of the video display, such as height, width, pixel depth, etc.

```
int vgaSetMode(int vgaMode, int shadowFlag)
```

vgaMode — A VGA Mode number. See the Table below for details.

shadowFlag — Controls the Second Sight's automatic shadowing of Apple II video modes. True (1) turns shadowing off, False (0) turns shadowing back on. If an application is going to draw directly to the screen, it must turn shadowing off.

Possible Errors:

vgaINVALID_MODE_NUM 0x01
An invalid video mode number was passed to vgaSetMode. The mode number given does not correspond to any supported video mode.

vgaUNSUPPORTED_MODE 0x02
The specified video mode is not supported by the monitor currently attached to the Second Sight. This could also mean that a > 256 color mode was selected, but the DAC is not capable of High Color or True Color.

```
int vgaSetPalette(RGB palette[], int startInd, int numEntries)
```

Sets `numEntries` entries of palette data, starting at entry `startInd` on the Second Sight card to the values specified in the pointer `palette`.

```
int vgaSetPaletteEntry(RGB *palEntry, int colorInd)
```

Sets a single palette entry specified by `colorInd` to the RGB values stored at the `palEntry` pointer.

```
int vgaLockPaletteRange(int startInd, int endInd, word userID)
int vgaUnlockPaletteRange(int startInd, int endInd, word userID)
int vgaFindAndLockPaletteRange(int numEntries)
```

Allocates a range of palette entries for the exclusive use of the calling application. This allows an application to set aside a certain number of colors for its own use, assured that its use of that entry won't interfere with other applications' colors or vice-versa.

```
int vgaMapColor(RGB *color);
```

Looks through the existing palette for the closest match to the specified color, and returns the index of that color. This routine may not return an index to a color anything at all like the specified one. If an application does not lock its colors, it may want to periodically redraw its windows after remapping colors with `vgaMapColor`.

```
int vgaUploadVideoData(void *dest_adr, LongWord size, void *gs_adr)
```

Similar to the _UploadData function in the low level library. However, vgaUploadVideoData only transfers data to video memory. Further, it knows how to deal with the 512K boundary. Use this function to seamlessly transfer data from the GS to video memory without worrying about splitting up your transfers - this routine does it for you.

`dest_adr` is of course the destination offset in VGA video memory. `size` is the number of bytes to transfer, and `gs_adr` is the address in GS memory of the beginning of the block to transfer.

File Formats

We highly recommend that only _one_ set of routines to read/write this file format be created: modules for Seven Hills' "Babelfish" product.

The requirements for an image file format for Second Sight are as follows:

- Must be able to handle images of varying pixel depths
- Should be somewhat compatible with existing software, to provide a "bridge" for users who do not yet have Second Sight.
- Should be quick to load and save.

To this end, we have chosen an extension to the APF (Apple Peferred Format). We have defined three new chunk types, "SVGA" to denote an 8bpp (or higher) image, "SVGC" to denote a PackBytes-compressed 8bpp image, and "SVGP", to denote an 8bpp palette.

SVGA

```
Word  pixDepth;          /* number of bits per pixel in this image */
Word  widthPixels;       /* number of pixels wide this image is */
Word heightPixels;       /* number of pixels tall this image is */
byte image[widthPixels*heightPixels*(pixDepth/8)];
                         /*
```

SVGC

```
Word  pixDepth;          /* number of bits per pixel in this image - always 8*/
Word  widthPixels;       /* number of pixels wide this image is */
Word heightPixels;       /* number of pixels tall this image is */
..scanline data..        /* each scanline in turn, run through PackBytes */
```

SVGP

```
Word  numEntries;              /* number of palette entries */
RGB palette[numEntries];       /* palette data in the form of 'RGB' struct */
```

To aid in compatibility of new pictures with existing applications, a grayscale version of the VGA image should be stored in the standard MAIN and PALETTE chunks of the file. Grayscale was chosen because of ease of conversion of any format picture to grayscale in 16-colors; if the application wishes to do a complicated quantization technique to choose colors for this "compatibility" image, it may do so, but this is not required.

Compression is not specified for 16bpp and 24bpp images, because the only decent existing standard for compression of TrueColor and HighColor images is JPEG - an algorithm that is extremely time-consuming to execute on a IIGS. However, we expect that JPEG will come into more common use on the IIGS due to SecondSight's TrueColor and HighColor capabilities.

Appendix A: VGA Video Modes

Text Mode	# Colors	Mode Number	Apple RGB Compatible?
40x25	16	$01	Yes
80x25	16	$03	Yes
80x43	16		No
80x50	16		No
80x60	16		No
132x25	16	$50	Yes
132x43	16	$51	No
132x60	16	$4F	No

Graphics Mode	# Colors	Mode Number	
320x200	256	$13	Yes
320x200	32K	$70	Yes
320x200	16M		
640x400	256	$61	Yes (interlace)
640x400	32K	$5B	Yes (interlace)
640x400	16M		
640x480	256	$53	No
640x480	32K	$5A	No
640x480	16M	$5F	No
800x600	256	$54	No
800x600	32K	$60	No
1024x768	256	$59	No

Emulation Mode	# Colors	Mode Number	
560x192	16	$FA	Yes
280x192	16	$FB	Yes
40x24	16	$FC	Yes
80x24	16	$FD	Yes
640x200	256	$FE	Yes

x256	256 colors using a 256 entry palette
x32K	65536 direct colors, using 6 bits for Green, and 5 each for Red and Blue, _or_ 32768 colors direct using 5 bits each for R, G, B.
x16M	16.7Million direct colors, using 8 bits each for Red, Green, and Blue

This list is not to be construed as a list of video modes that any particular application or any particular version of the C VGA library suports.