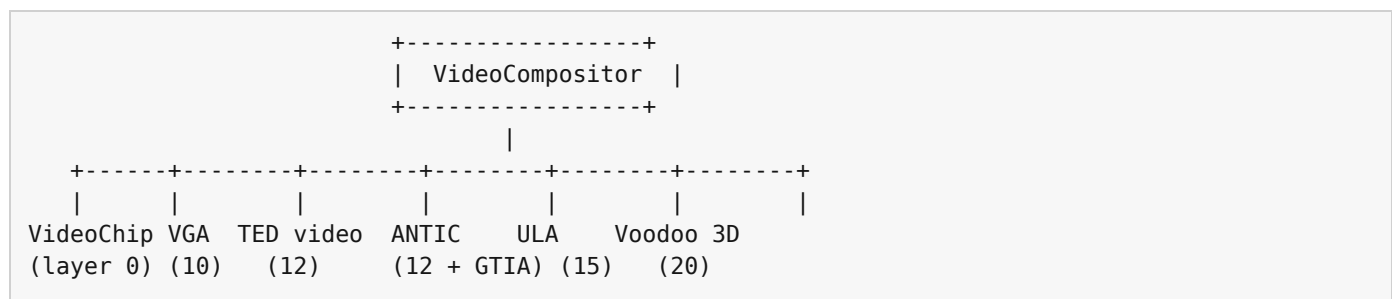


Chapter 3 - Display Model Overview

Intuition Engine has six picture-making chips on the same machine. Each one is a display card on the shared bus. It draws a frame in its own format and gives that frame to a single piece of hardware called the **VideoCompositor**. The compositor stacks the six frames on top of each other and sends the result to the screen.

This chapter explains how the stack works, what the output looks like, and how colours are chosen. Each of the six chips has a chapter of its own that tells you how to program it directly.

3.1 The six picture sources



Each source produces a frame at its own size and resolution. The compositor accepts whatever size each source produces and scales it to fit the output screen. The output screen is 1920 pixels wide and 1080 pixels tall.

Chip	Best for	See chapter
VideoChip	Bitmap and CLUT8 modes, copper list, blitter	4
VGA	Text and 256-colour graphics modes	5
TED video	TED-class character and bitmap modes	6
ANTIC	Display-list video with GTIA companion	7
ULA	ZX-class attribute display	8
Voodoo 3D	Triangle rasterisation with Z-buffer and texture	9

You may use any of the six at the same time. You may also leave some of them switched off. A source that is disabled or that returns no frame contributes no pixels. The question is not which computer you are using, but which display card should own this part of the final picture.

3.2 The layer stack

Every source has a fixed **layer number**. The compositor sorts the sources by layer number, smallest first, and draws each one into the output frame in that order. A later source paints on top of an earlier one.

Layer	Source
0	VideoChip
10	VGA

Layer	Source
12	TED video
13	ANTIC
15	ULA
20	Voodoo 3D

The layer numbers are part of the hardware. They cannot be changed from BASIC. To put a frame behind everything, use the VideoChip (layer 0). To put a frame on top of everything, use Voodoo 3D (layer 20).

The gaps between the numbers (10 to 20) leave room for future sources without disturbing the existing order. Today only the six sources above exist.

3.3 Mask blending

The compositor does not mix colours. It treats each source pixel as either **opaque** or **transparent**. A pixel is opaque if its alpha byte is non-zero. A pixel whose alpha byte is zero is also treated as opaque if any red, green, or blue byte is non-zero; in that case the compositor promotes the pixel to full alpha before drawing it. The only transparent direct-colour pixel is all zero.

Source pixel	What the compositor does
\$00000000	Leave the destination pixel as it was.
non-zero alpha	Replace the destination pixel with the source's RGB.
zero alpha, non-zero RGB	Replace the destination pixel and force alpha to \$FF.

The result is the same as if you had laid one transparent acetate on top of another and looked at the stack from above: the topmost opaque pixel at any position is the one you see. Each chip's colour-zero or background pixel is normally transparent, so the chip below shows through where the chip above is "blank".

The compositor never partially blends two pixels. If you want something that looks like a half-transparent overlay, write that into the source's pixels directly - for example by drawing every other pixel.

3.4 Output resolution and refresh

The compositor sends the final frame at the following fixed rate:

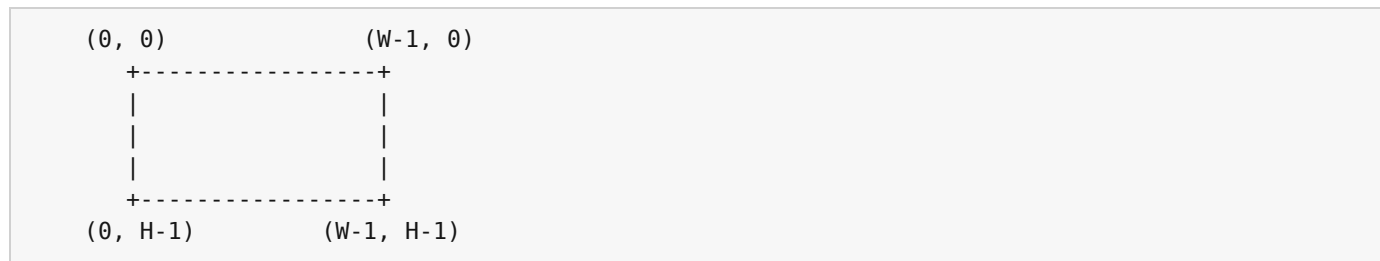
Property	Value
Output width	1920 pixels
Output height	1080 pixels
Frame rate	60 Hz
Pixel format	RGBA, 4 bytes per pixel

All display cards are advanced from the same 60 Hz frame cadence. That does not make every chip the same size or shape. It means the compositor has one regular moment to collect finished frames, apply layer order, and publish the final picture. Chips with scanline features, such as the VideoChip copper or ANTIC display lists, still do their own per-line work before their finished frame reaches the stack.

A source that produces a frame at the full 1920×1080 resolution is copied to the output 1:1. A source that produces a smaller frame is scaled up by the compositor. The scale mode is **stretch-fill**: the source frame is rescaled to fill the entire output area, even if the aspect ratios differ.

3.5 The coordinate system

Every chip uses the same coordinate convention:



The origin is the top-left corner of the picture. X increases to the right; Y increases downward. W and H are the width and height of the source's frame, which may be smaller than the output screen.

3.6 The palette

Each chip has its own palette. There is no global palette shared between chips. A palette entry is a 24-bit value made of three 8-bit components for red, green, and blue.

From BASIC, the PALETTE and PLOT statements operate on the VGA chip (see Chapter 5). The other chips are programmed through their own MMIO registers - either with POKE32 from BASIC or directly from machine language. Each chip's chapter lists the addresses.

The number of palette entries available depends on the chip. The typical sizes are:

Chip	Palette size
VideoChip	256 entries (CLUT8 mode)
VGA	256 entries
TED video	121 unique colours (16 hues \times 8 luminances + black)
ANTIC	256 entries (16 hues \times 16 luminance steps)
ULA	15 unique colours (8 base + 7 bright; black has no bright form)
Voodoo 3D	Per-vertex 24-bit colour, plus a 256-entry texture palette for indexed textures

See the chip's chapter for the exact byte layout of its palette and for any chip-specific extensions (such as ULA attribute pairing).

3.7 The vertical retrace

After the compositor finishes assembling one output frame, there is a short pause before the next one begins. This pause is the **vertical retrace**. It is the safe time to update display memory: any changes you make during the retrace are picked up by the next frame as a single update, with no tearing.

The VSYNC statement waits for the next vertical retrace:

```
10 VSYNC
20 REM update display memory here
```

Every chip exposes its own retrace status flag in its MMIO map. See Appendix D for the addresses.

3.8 A first display program

This short program uses the VGA source because it is the quickest way to draw from BASIC. It selects 320 x 200 indexed-colour graphics, loads three palette entries, draws diagonal lines, and waits for the next retrace before returning to the prompt.

```
10 REM DISPLAY MODEL FIRST PICTURE
20 SCREEN &H13
30 PALETTE 1,63,0,0
40 PALETTE 2,0,63,0
50 PALETTE 3,0,0,63
60 CLS 0
70 FOR I=0 TO 199
80 PLOT I,I,1
90 PLOT 319-I,I,2
100 NEXT I
110 FOR X=0 TO 319
120 PLOT X,100,3
130 NEXT X
140 VSYNC
```

The two diagonal lines show the top-left origin and downward-growing Y coordinate. The horizontal line shows that colour comes from the VGA palette entry selected by the pixel value. VSYNC gives the next frame a clean point at which to appear.

3.9 Choosing a chip

You normally pick a chip first and then program it for the rest of the picture. Reasons to pick each one:

- **VideoChip** is the IE's general-purpose framebuffer chip. It handles 256-colour CLUT8 bitmaps and direct-colour RGBA bitmaps, a copper list for per-scanline changes, and a blitter for hardware copies. Pick it when you want maximum flexibility and a programmable raster.
- **VGA** is the cleanest text mode and the smoothest path to a scrolling 256-colour bitmap with rectangular tiles. Pick it for text adventures, scrolling shoot-em-ups, and anything that wants hardware character generation.
- **TED video** mirrors the C16/Plus-4 picture model and lets you reproduce that machine's look directly in hardware.
- **ANTIC + GTIA** mirrors the Atari 8-bit picture model. The display list lets you change the mode line by line and is the shortest path to colourful mixed-mode screens.
- **ULA** mirrors the ZX Spectrum picture model: a monochrome bitmap with 8×8 colour attributes.
- **Voodoo 3D** rasterises triangles into a Z-buffered RGBA frame. Pick it for 3D scenes.

You can layer two chips for tricks the compositor allows for free: draw a bitmap on the VideoChip and place text from the VGA on top, or use the ULA to overlay an attribute grid on an ANTIC display. Because all six cards feed the same compositor, the final picture is still one Intuition Engine screen. The compositor handles the stacking; you do not write any blending code yourself.

3.10 What comes next

The remaining chapters of Part II describe each chip in turn. Each chapter follows the same plan:

1. What the chip's frame looks like (mode list, resolution).
2. The MMIO map.
3. How to set a mode and draw a pixel.
4. The chip's hardware features (sprites, scrolling, copper, etc.).
5. Example BASIC programs.

Appendix D is the consolidated MMIO map for every chip. Appendix K is a block diagram of the whole video system.