The NGT31 is an easy to use, 5v compatible propeller graphics device. It communicates simply with UART communication.

It will help you make your project by featuring:

- An Arduino shield-compatible interface
- Supports 5v and 3.3v interface
- Does basic 3D graphics efficiently
- Pins used on Arduino can be configured
- Quick Sprite definition and display
- Easy to modify the NGT31's firmware
- VGA without costing tons of RAM, program space, and CPU time

Table of Contents

Conventions

- Command arguments sent to the NGT31 are in Big Endian format.
- Data loaded into VRAM is in Little Endian format.
- A byte is 8 bits
- A word is 2 bytes
- A long is 4 bytes

NGT31 Overview

What can it do?

- The NGT31 produces VGA video output
- Draws sprites
- Any basic graphics function
- Basic accelerated 3D drawing
- Can do rainbow colors in foreground or background across tiles
- The firmware is open-source, so if you like you may modify your NGT31's firmware
- Can be used as an arduino shield, or hacked in various ways

How does it work?

The NGT31 uses the **parallax propeller,** an 8-core microcontroller, clocked at 80 MHz. It is excellent for producing video signals, which is why it was chosen to make the NGT31. It runs at 3.3v, so it cannot directly connect to a 5v arduino, Z80 UART, or other 5v device, so the NGT31 uses a **voltage level shifter.** It **uses multiple cores** to generate video. It communicates with the host device with **115200 bps UART.** If you modify the firmware, or otherwise reprogram the NGT31, the way that it communicates can change.

What sets NGT31 apart from other graphics shields?

- Accelerated 3D color graphics
- Supports the quick repeated drawing of sprites, as opposed to pixel-by-pixel draws
- Internal RAM stores details about your drawing operations, so less communication is needed, and offloads the main computer's resource demands
- Compatible with more host devices than just an arduino, anything capable of communicating on 115200 bps UART at 3.3v-5v will be compatible.
- Advanced arduino library for graphics and creating sprites at runtime
- Can draw rainbow text and sprites
- Easily reprogrammed with a Prop Plug, as opposed to FPGA or CPLD versions

In short, it is an **easy to use**, **VGA outputting**, **expandable** and **reprogrammable** propeller graphics device.

How To Use The NGT31 With Arduino

Here begins directions on how to get off the ground with your new NGT31 board.

Connecting the NGT31 to your Arduino

A note on communication voltages:

- Since the serial output from the NGT31 is always 3.3v, regardless of what TX pin it's from, **3.3v-compatible devices will never have problems connecting to the NGT31 from voltage incompatibilities. This guarantees compatibility with 3.3v arduino boards.**

Stack the NGT31 onto the Arduino, with the VGA connector the opposite way of the arduino's USB and Barrel Jack connectors.

For interface with the NGT31 via arduino boards the NMT_GFX Arduino library can be used. It is best to have the library. Download it from my GitLab here.

To initialize the NGT31 it the library must be set up first:

```
#include <NMT_GFX.h>
NMT_GFX ngt;
```

Easily done, simply includes NMT_GFX and creates an object to use. Now to actually initialize the NGT31:

```
ngt.begin(11,10);
```

ngt.begin properly resets the NGT31 so it is ready to receive commands. Now the NGT31 is initialized.

Colors and the NGT31

It is important to understand how colors work specifically for the NGT31. Please read this section to understand.

The NGT31's bitmap is divided into tiles. The default NGT31 firmware has 18 x tiles and 16 y tiles. Each tile has 16 columns and 16 rows. Each tile also has only 4 colors. These colors must be selected from one of 64 ‘color slots’. To make this concept easier to understand, here is an example:

Tile X is currently using color slot zero. There is text on the tile with the foreground using color 2 of the color slot’s 4 colors. By changing color slot zero’s color 2, that text’s color changes for all tiles using that color slot, including tile X. If tile X’s color slot is changed to slot 5 for instance, all the pixels in tile X will adopt their corresponding colors in slot 5.

Although this could seem a disadvantage, it can be useful for making a sprite change color as it moves or making rainbow text. It also allows the background of certain tiles to change, or text to change color at certain boundaries. There are both pros and cons to this approach to displaying colors.

List of arduino library functions

The NMT_GFX Arduino library is available here.

`NMT_GFX::NMT_GFX();`
Starts the NMT_GFX library with NGT20-compatible configuration.

`NMT_GFX::NMT_GFX(rx, tx);`
Starts the NMT_GFX library with the specified RX and TX pins.

`void NMT_GFX::print(x);`
Print character or string x at the cursor and advance the cursor once.

`void NMT_GFX::println(x);`
Print the string or char x onto the screen and advance the cursor to the end of the string, then go to the next line.

`void NMT_GFX::write_at(q, int x, int y);`
Print the string or char q onto the screen where the upper left of the string is placed at (x, y). Does not affect cursor position.

`void NMT_GFX::set_cursor_pos(int x, int y);`
Set cursor position to (x, y). Note that unlike other functions, x and y are measured in characters, which are 6x13 pixels each.

`void NMT_GFX::line(int x1, int y1, int x2, int y2);`
Draw line from (x1, y1) to (x2, y2). (x2, y2) is new endpoint.

`void NMT_GFX::box(int x1, int y1, int x2, int y2);`
Make a square between (x1, y1) and (x2, y2)

`void NMT_GFX::fill_box(int x1, int y1, int x2, int y2);`
Make a solid square between (x1, y1) and (x2, y2)

`void NMT_GFX::fast(int x, int y);`
Makes a line between the endpoint of the last line or the last pixel, whichever occured last, and (x, y). New endpoint is (x, y)

`void NMT_GFX::clear();`
Fill every tile with color zero in its color slot.

`void NMT_GFX::fill(byte color);`
Fill every tile with color in its color slot.

`void NMT_GFX::set_color(byte color);`
Make next draws use color.

`void NMT_GFX::pixel(int x, int y);`
Draw pixel at (x, y)

`byte NMT_GFX::x_tiles();`
Returns number of x tiles. Multiply this by 16 to get number of columns.

`byte NMT_GFX::y_tiles();`
Returns number of y tiles. Multiply this by 16 to get number of rows.

`char* NMT_GFX::get_card_ver();`
Returns string for device model. For an NGT31 it should be “NGT31”. Newer or older versions may say something different, like “NGT20”.

`void NMT_GFX::block_color(byte slot, byte color);`
Set color (slot>>6) in slot (slot&63) to the argument ‘color’.

`void NMT_GFX::tile_color(int tile, byte slot);`
Make a tile use a specific color slot.

`byte NMT_GFX::make_color(byte r, byte g, byte b);`
Returns a color for use in `NMT_GFX::block_color()` using RGB color input.

`void NMT_GFX::translatef(long x,long y, long z);`
Moves the camera to the coordinates (x,y,z). Does not render a new frame!

`void NMT_GFX::rotatef(uint16_t yaw, uint16_t pitch);`
Sets the camera yaw and pitch. Does not render a new frame!

`void NMT_GFX::point3d(long x, long y, long z, uint8_t color);`
Draws a 3D point at the given location

`void NMT_GFX::line3d(long x, long y, long z, uint8_t color);`
Draws a line from the last drawn point or line, to the specified 3D point.

`void NMT_GFX::sprite3d(long x, long y, long z, uint16_t handle, uint8_t rotation);`
Draws a sprite at the 3D coordinates specified

`void NMT_GFX::exec3d(uint16_t handle, uint16_t numCommands);`
Executes many 3D commands from VRAM, starting at the VRAM address 'handle', and it will execute numCommands 3D commands. It is usually better and easier to use an Object3D object to do this, instead of calling 'exec3d' directly. An example to use Object3D comes with the library, and there is also an example on the wiki:
http://wiki.nuclearman.technology/mediawiki//index.php/NGT31#3D_Example_and_Explanation

There are code examples for many of these in the NMT_GFX library.

Creating, loading, and displaying a sprite definition

All examples here are written in C++ for the Arduino. They use the NMT_GFX Arduino library available here.

All the code I present here will be in the Arduino function "setup", except for variable declarations.

This example uses NMT_GFX's Sprite class. It is a simple and easy to use class that only requires the definition of the sprite size and center, some RAM to use, and some initialization. First some memory must be allocated for the Sprite:

```
byte image[52];
```

This is enough space for an 8x24 sprite. 4 bytes are used for parameters and 48 bytes times 4 pixels/byte = 192 pixels. X length has to be a multiple of eight in RAM, so 192/8 = 24 y pixels. Ok, here is where the sprite is defined:

```
Sprite sprite;
```

No fancy constructor, just the definition. It is initialized by these lines:

```
// tell sprite where to store its data
sprite.binary_image=(byte*)image;
// tell sprite how big it is
sprite.set_size(8,24);
// set the center at 4,4 pixels
sprite.set_center(4,4);
```

The sprite is told the location of its memory, its size, and its center. Now it is ready to be written.

The writes are done with function calls to Sprite::fill() and Sprite::pixel()

```
// fill sprite
sprite.fill(2);
// draw some pixels
sprite.pixel(3,3,0);
... And more writes ...
```

sprite.fill(2) makes all pixels in the sprite color 2 of whichever color slot that pixel occupies. The color slot occupied is the pixel's tile's color slot. To understand how these colors work see Colors and the NGT31.

sprite.pixel(3, 3, 0) makes the pixel at (3, 3) color 0 of its color slot.

Once all writes are done on the sprite it MUST be uploaded to the NGT31:

```
sprite.upload();
```

Easy. Now we can put it across the screen a ton of times:

```
// Display sprite at 50,60 at a 0 degree angle
sprite.display(50,60,0);
sprite.display(20,60,0);
sprite.display(60,60,0);
```

These lines display the same image 3 times, each time minimal exchange happens between the NGT31 and the Arduino, as opposed to slowly and painfully writing raw pixels and boxes to the screen, in a repetitive inefficient process.

sprite.display()'s 3rd argument specifies the orientation of the sprite. 0 is 0 degrees, 1 is 90 degrees, 2 is upside down, and 3 is 270 degrees.

Now things get even cooler though. The array of bytes given to the Sprite is an image and can be saved somewhere and reloaded later. To reload a Sprite from a raw memory/binary image simply do this:

```
sprite.binary_image=your_saved_image_as_byte_array
```

Now you can save a sprite in eeprom, on an SD card, and more, then reload it later.

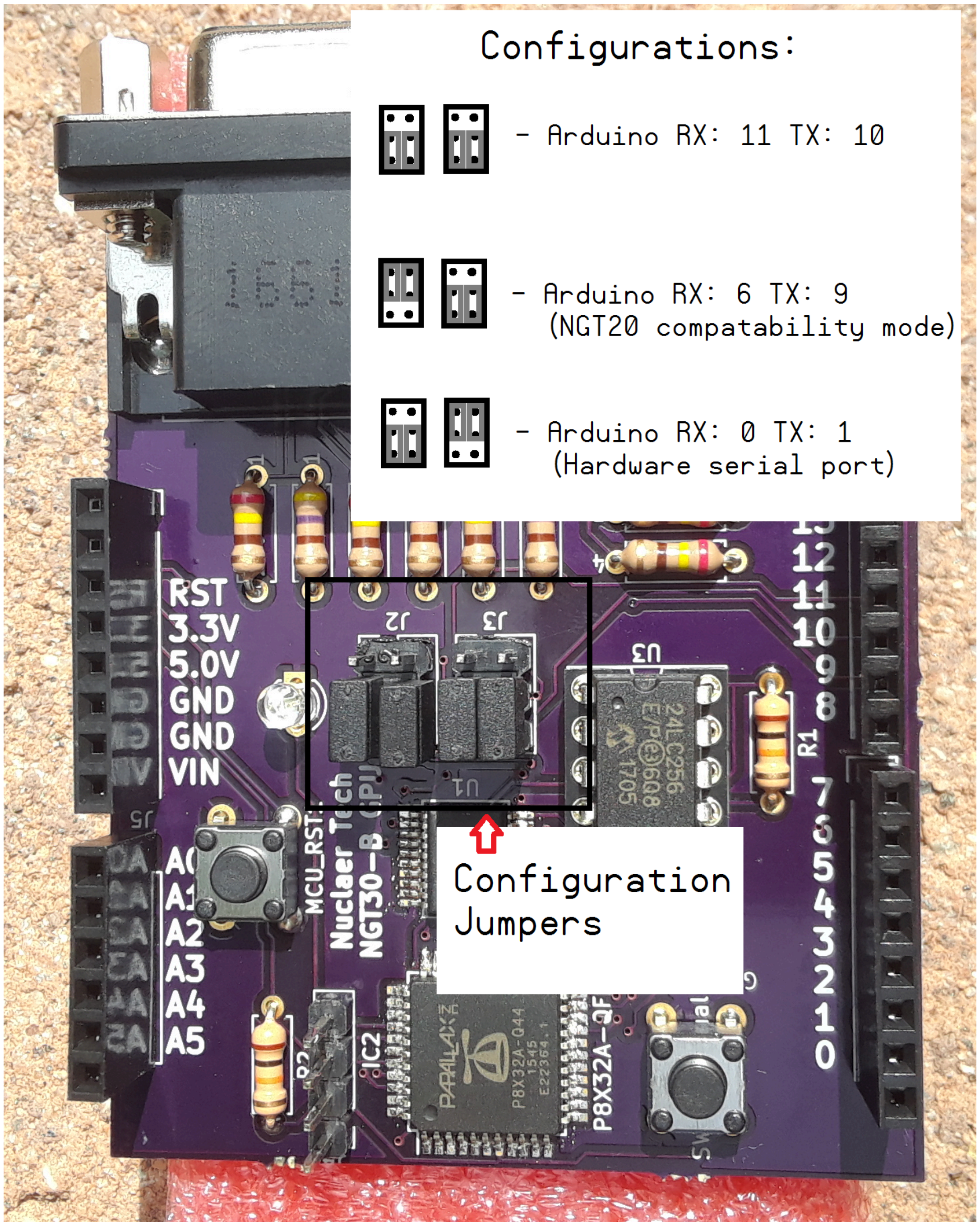
Rendering in 3D

This information is now kept on the wiki:
http://wiki.nuclearman.technology/mediawiki//index.php/NGT31#3D_Example_and_Explanation

The NGT31's 3D architecture is quite different from previous NGT* shields, and the 3D code is not compatible.

Configuring pin connections.

The NGT31 can have its pins connected to different Arduino pins with jumpers on the board to avoid pin conflicts.



Troubleshooting

NGT31 won’t display anything. TV/Monitor gets signal.

The NGT31 may be configured to use different pins than the Arduino is expecting, or the code for the Arduino is connecting on the wrong RX/TX pins. Typically, the NGT31 has RX and TX pins configured as 11 and 10 respectively, however this can be adjusted with the jumper wires on the NGT31.

NGT31 was working, then the green light turned solid and it stopped responding.

This will happen if an invalid sprite is displayed. Simply hard reset the NGT31 to set it to its original state.

Colors and the NGT31

Please read this for proper understanding of how colors work for the NGT31.

The NGT31’s bitmap is divided into tiles. The default NGT31 firmware has 16 x tiles and 12 y tiles. Each tile has 16 columns and 16 rows. Each tile also has only 4 colors. These colors must be selected from one of 64 ‘color slots’. To make this concept easier to understand, here is an example:

Tile X is currently using color slot zero. There is text on the tile with the foreground using color 2 of the color slot’s 4 colors. By changing color slot zero’s color 2, that text’s color changes for all tiles using that color slot, including tile X. If tile X’s color slot is changed to slot 5 for instance, all the pixels in tile X will adopt their corresponding colors in slot 5.

Although this could seem a disadvantage, it can be useful for making a sprite change color as it moves or making rainbow text. It also allows the background of certain tiles to change, or text to change color at certain boundaries. There are both pros and cons to this approach to displaying colors.

Technical Specifications

Communication protocol of the NGT31

The NGT31 accepts commands on the UART port and interprets them. It then returns a result on the UART. Some commands take arguments to give the NGT31 more information about what it is being told to do.
Note that:

- All commands are sent in sequences of bytes
- String arguments are NOT zero terminated, but CR-terminated.
- Any unlisted command IDs have no effect on the NGT31. For a more advanced device, the effect will be different and that device's datasheet should be referenced.

To run a command on the NGT31, first a command ID must be sent, then its arguments. The arguments are in the same order as specified in the table under "Commands At A Glance".

Argument Data Types

Different arguments are different data types.
Data types include:

- BYTE
- WORD (2 bytes)
- STRING (any number of bytes, CR-terminated)

Any command with arguments will accept more data after the sending of the command ID. For each argument data must be supplied, the length of each argument depending on data type. Refer to the bulleted list above, and remember to CR-terminate all STRINGs.

Refer to the table below for the arguments of each command.

Commands At A Glance

Command ID	Command Description	Arguments
13	Newline	None
32	Get device type (CR-terminated)	None
48	Reset NGT31	None
49	Clear Screen	None
50	Set Cursor Pos	BYTE x, BYTE y

51	Write string	STRING str, BYTE 0x0D
52	Draw line	WORD x1, WORD y1, WORD x2, WORD y2
53	Fill Screen with color	None
54	Fill box	WORD x1, WORD y1, WORD x2, WORD y2
55	Write string at position	WORD x, WORD y, STRING str, BYTE 0x0D
56	Write pixel	WORD x, WORD y
57	Set Color	BYTE color
61	Change color slot	BYTE slot, BYTE color
62	Draw sprite	WORD x, WORD y, BYTE rotation, WORD address
63	Write RAM	WORD address, BYTE data
64	Get output type	None
66	Set tile color	WORD tile, BYTE slot
67	Fast line	WORD x, WORD y
100	Render 3D Object	WORD address, WORD numOperations
101	Get Graphics Memory Length	None
102	Set camera position	LONG x, LONG y, LONG z
103	Set camera rotation	WORD yaw, WORD pitch
107	Change baud rate	LONG baud
108	Set Draw Mode	BYTE draw_mode
109	Move Sprite	WORD x1, WORD y1, WORD x2, WORD y2,
110	Bulk write VRAM	WORD start_address, BYTE num_bytes, BYTES data
111	Wait for VSYNC	None
112	Draw 3D Point	LONG x, LONG y, LONG z, BYTE color
113	Draw 3D Line	LONG x, LONG y, LONG z, BYTE color
114	Draw 3D Sprite	LONG x, LONG y, LONG z, WORD address, BYTE rotation

Commands - In Depth Descriptions

Please refer to the table above for information on command arguments.

Command 13 : New Line

Will advance the cursor downward 1 character. If the cursor leaves the screen then the entire screen is scrolled up. This affects all pixels on the screen, so be careful or use this to your advantage.

Command 32 : Get Device Type

Will send a null terminated string indicating the device type. For the NGT31 it would literally be “NGT31”, then a NULL.

Command 48 : Reset

Will completely reboot the NGT31, all pixels, sprites, and colors will be lost. Equivalent to pressing the reset button, except Command 48 requires no mechanical action.

Command 49 : Clear screen

Sets cursor to top left and makes every pixel use it’s tile’s zeroth color. Note that this means pixels in different tiles may have different colors due to the tiles having a modified tile colors. Use Command 61 to change a color set’s colors and Command 66 to change a tile’s color set.

Command 50 : Set Cursor Pos

Set cursor position for Command 51. Measured in characters. Each character is 6 pixels wide and 13 pixels tall.

Command 51 : Write string at cursor position

Writes a carriage-return terminated string on the screen and advances the cursor to after the string’s end. If the string goes past the end of the screen it loops back on the other side. If it continues below the screen’s lower edge then the screen scrolls up, including all pixels, and the string continues writing. BE SURE THE STRING IS CR-TERMINATED!

Command 52 : Draw Line

Draws a line from (x1, y1) to (x2, y2).

Command 53 : Fill Screen with color

Fill screen with color. Note that color is NOT an RGB color, but a number from 0-3 referring to tile color. To change the color slot of a tile or colors in a color slot use commands 61 and 66.

Command 54 : Fill box

Fill box with opposite edges at (x1,y1) and (x2, y2)

Command 55 : Write string at position

Writes string at (x, y)
Does not change cursor position. X and y are in pixels, not characters. Also note that the justification of the text places the upper left of the string at (x, y).

Command 56 : Write Pixel

Draws a pixel at (x, y)

Command 57 : Set Color

Set color for next draw/text commands.   Note that color is NOT an RGB color, but a number from 0-3 referring to tile color. To change the color slot of a tile or colors in a color slot use commands 61 and 66. Does not apply to command 49 (Clear Screen)

Command 61 : Change color slot

There are 64 color slots. The lower 6 bits of the slot argument define which color slot. The upper 2 bits select one of 4 of the 1-byte color definitions for that slot.

Command 62 :Draw sprite

Draws a sprite at (x, y). Rotation is 0-3, where 0 is no rotation, 1 is 90 degrees, 2 is 180, and 3 is 270. Sprite is the offset of a sprite definition written to the NGT31's RAM. For instructions on creating a sprite definition, loading it, and displaying it refer here.

Command 63 : Write RAM

The NGT31 has 2 KiB of RAM that can be written to store sprites. You could store about 255 8x4 pixel sprites with this space. Command 63 writes one byte of this memory for loading sprites to the NGT31. For instructions on creating a sprite definition, loading it, and displaying it refer here.

Command 64 : Get output type

This command returns data about your NGT31 depending on its firmware. The command sends back 3 bytes of data. The first byte is the number of x tiles. Every X tile has 16 columns. The second byte is the number of y tiles. Every Y tile has 16 rows.

The last byte should be 'D' if the NGT31 has the default firmware. If the firmware does not support NTSC output the char will be 'V'. If it only supports NTSC the char will be 'N'.

Command 66 : Set tile color

Sets specified tile to use specified color slot.

Command 67 : Fast line

Excellent for drawing lines fast - takes about half the time of a normal line draw.

If a line was drawn last then it's endpoint becomes the fast line's starting point. If it was a pixel then that pixel becomes the starting point. The fast line's endpoint is (x, y).

Command 100 : Render 3D Object

Executes numCommands 3D commands in VRAM from the address specified. This allows large 3D objects to be drawn with only one call.

Command 101 : Get Graphics Memory Length

Upon executing this command the NGT31 will send back a word which is the number of kilobytes (1024 bytes, not 1000 bytes) that command 63 has available to write to.
This command can be used to check if it is safe to write to an address of VRAM.

Command 102: Change Camera Position

Accepts 12 bytes of data, as three 4-byte integers in big-endian encoding. Integers are supplied in the order x, y, z.

Command 103: Set Camera Rotation

Accepts 4 bytes of data, as two 2-byte shorts, in big-endian format. The first short is the new camera yaw, and the second short is the new camera pitch.

Command 107: Change Serial Baud Rate

This command accepts one big-endian 4-byte integer as a parameter. This integer encodes the new baudrate to use. Note that providing a bad baud rate may cause communication problems, which would prevent changing the baudrate back without a physical reset. This command waits 0.1 seconds after changing the baudrate, then the command's acknowledge is sent at the new baudrate. This wait gives the host processor time to flush the buffer and change its own baudrate. For AVR devices, the software serial port does not seem to work above 115200 baud. If higher baudrates are desired, the hardware serial port must be used.

Command 109: Move Sprite

Draws a sprite at (x2, y2), after clearing the sprite previously drawn at (x1, y1). Rotation is 0-3, where 0 is no rotation, 1 is 90 degrees, 2 is 180, and 3 is 270. Sprite is the offset of a sprite definition written to the NGT31's RAM. For instructions on creating a sprite definition, loading it, and displaying it refer here.

Command 110 : Bulk Write RAM

Writes num_bytes bytes of memory at the specified RAM address. For instructions on creating a sprite definition, loading it, and displaying it refer here.

Command 111: Vsync

Waits for a video period to end.

Command 112: Draw 3D Point

Computes the screen coordinates from x, y, and z, then draws a pixel of the specified color at that location. The screen coordinates are saved for any subsequent calls to command 113 (draw 3D line) or to command 67 (quick 2D line)

Note that nothing is drawn if the point is out of the camera's view.

Command 113: Draw 3D Line

Computes the screen coordinates from x, y, and z, then draws a line of the specified color from the new screen coordinates to the last saved screen coordinates. The new screen coordinates are saved for any subsequent calls to command 113 (draw 3D line) or to command 67 (quick 2D line).

Note that nothing is drawn if the point is out of the camera's view.

Command 114: Draw 3D Sprite

Computes the screen coordinates from x, y, and z, then draws the specified sprite at that location. Note that the sprite's size, rotation, and shear are not adjusted by the camera/world transform.

Note that nothing is drawn if the point is out of the camera's view.