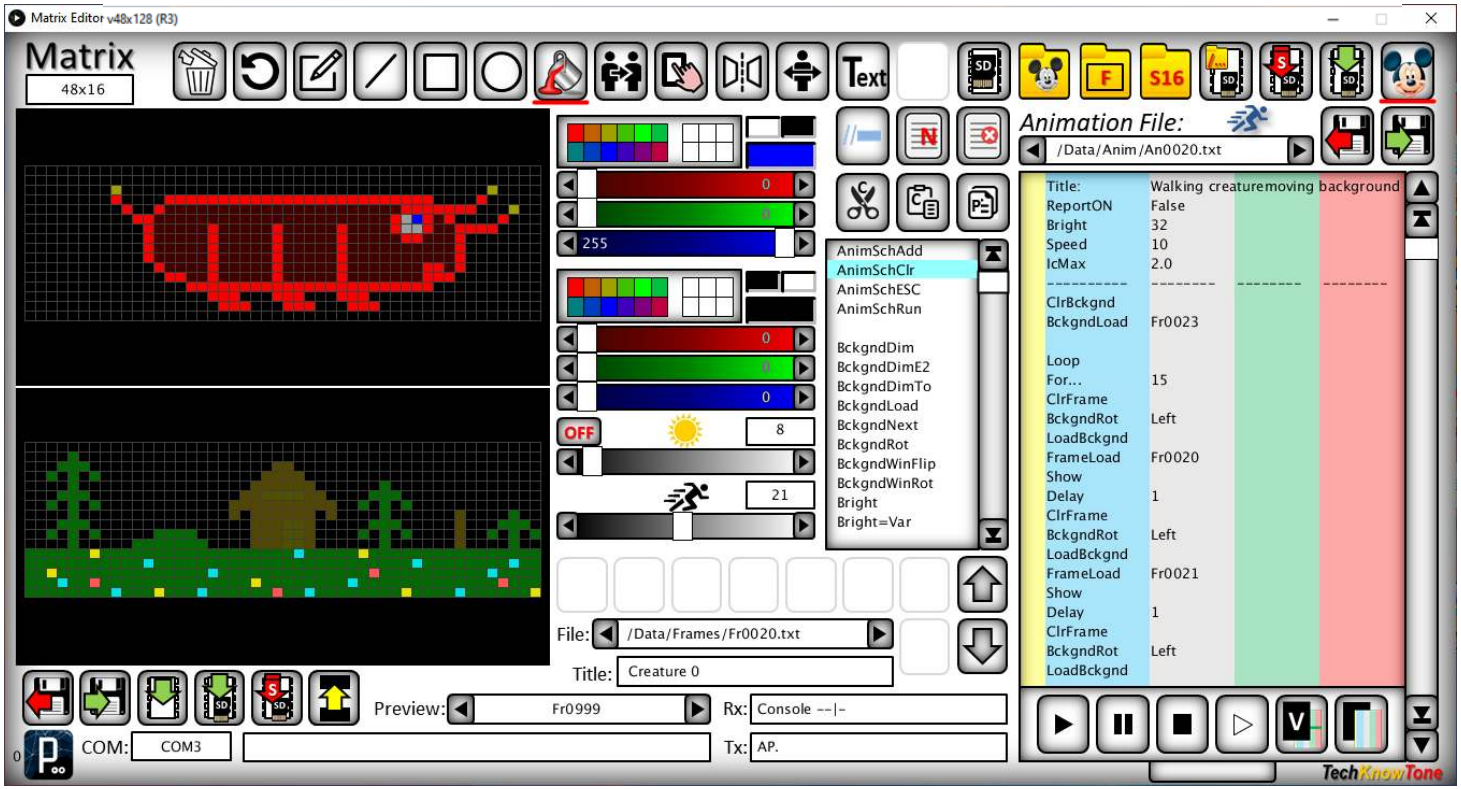
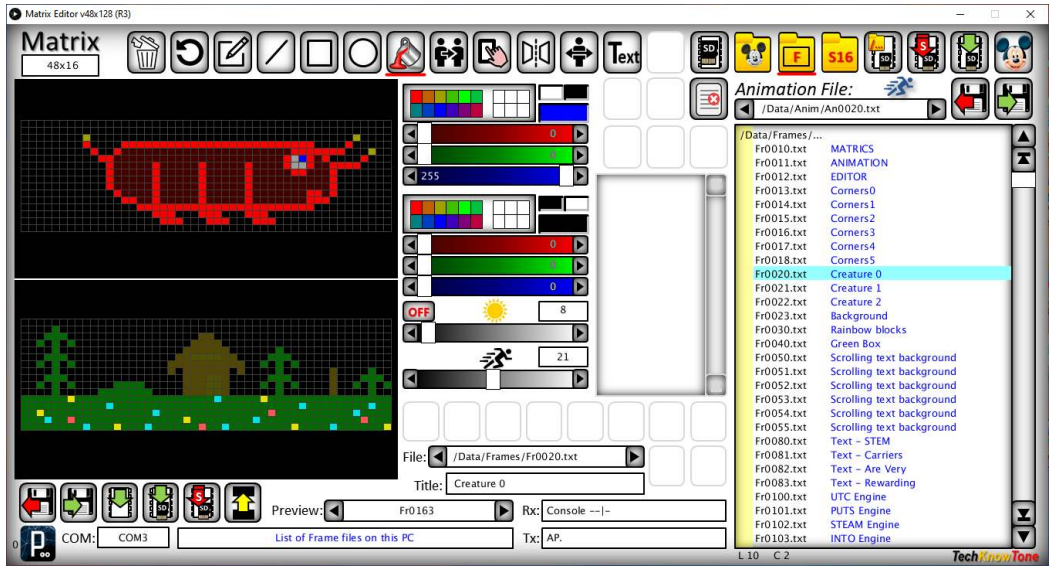
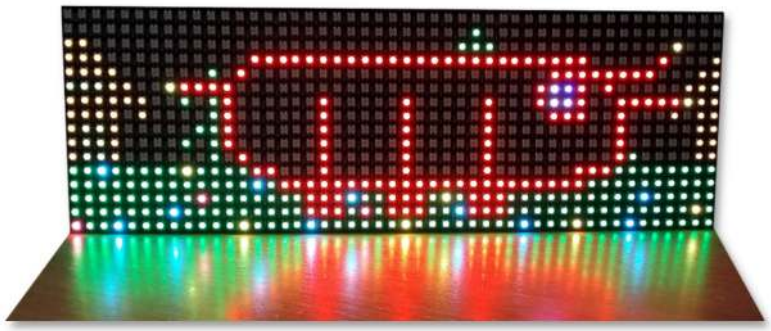
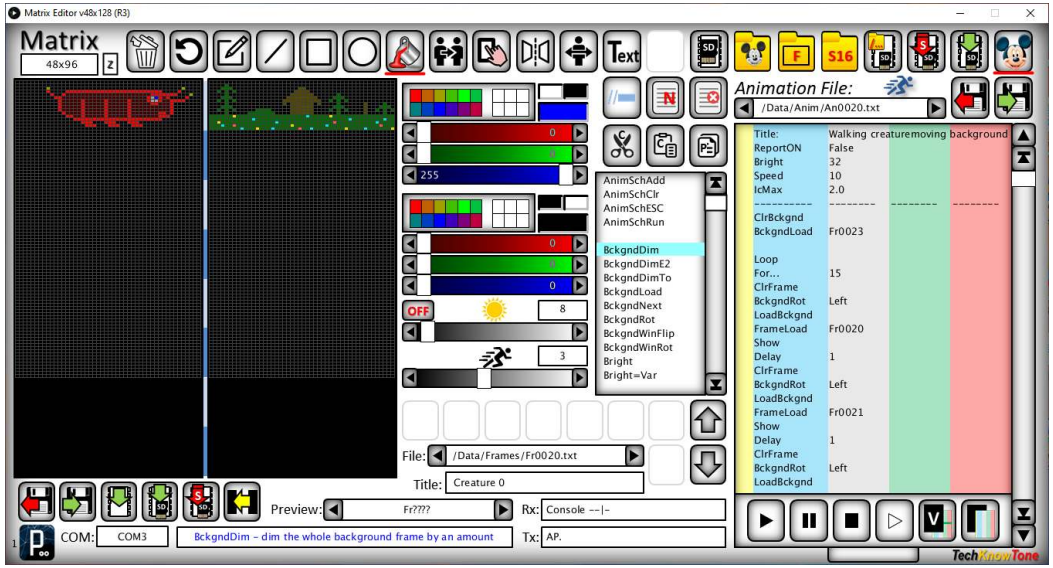
MATRIX
Animation Editor
Quick Guide (R3)

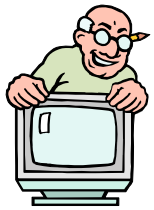


Introduction



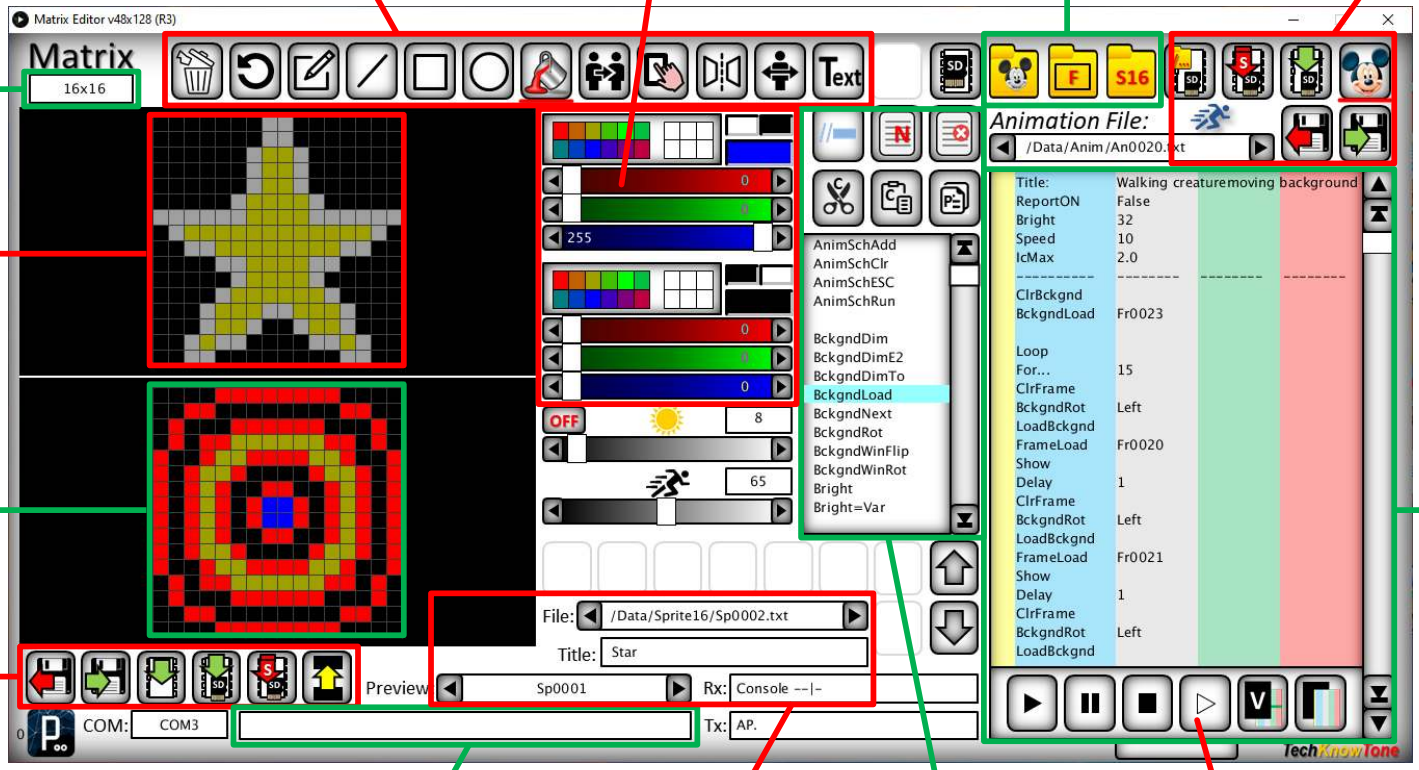






Matrix is a Windows app, designed for use with WS2812B RGB LED panels to perform two functions, when used in conjunction with a suitable microcontroller. It enables you to design colourful images, and to bring them to life by creating animation scripts, which are all stored in files on a microSD card. A number of LED panel configurations are supported, ranging from a single 16x16 LED panel, up to a 3x8 panel 48x128 LED array. This simple guide shows you the key features of the Matrix app, which should be sufficient to get you started. Matrix is written in C++ using the Processing development platform, compiled into Java 32-bit, to run on Windows based PC's, with Java pre-installed, and a minimum screen resolution of 1366 x 768 pixels.

Functions



Drawing tools

Colour Palettes

Folders

Animation File Access

Frame Size

Editor Window

Preview Window

Multi-view List Window

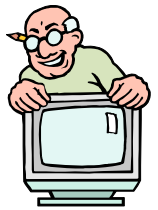
File access

Help System

File paths

Animation Script Tools

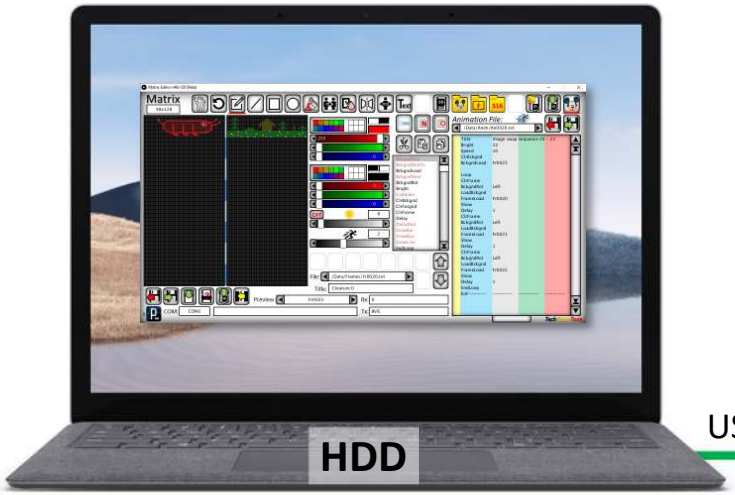
Animation 'Play' Tools



The left hand side of the Matrix app is used for creating graphics, whilst the right hand side is used for creating animation scripts and viewing file lists. The '**Help**' system field at the bottom of the app displays key information about items under the mouse pointer. Try moving the mouse pointer around to view the help messages; you can learn a great deal from reading these prompts.

Note that some items only appear under certain conditions; like the additional controls for the animation script editor. The Preview window also acts as a source for colour picking, and copying graphics and cloning from other images.

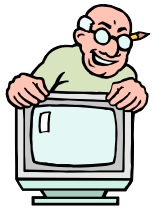
File Storage



USB

App root/Data
- /Anim — Text scripts
- /Frames — Graphics
- /Sprite16 — Graphics 16x16

SD root/Data
- /Anim — Numeric scripts
- /Frames — Graphics
- /Sprite16 — Graphics 16x16



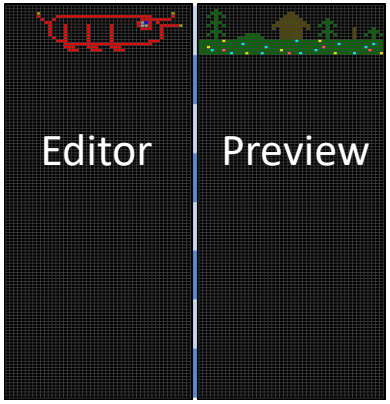
Matrix creates files for graphics and animation scripts, which are stored on your PC, and also transferred to your micro as data, over the serial USB link, and stored on its microSD card. The microSD file system is based on FAT32, hence filenames are limited to 8 characters, plus a 3 character extension; like ‘FR0000.txt’. For consistency this naming convention is also used for PC filenames.

The data storage format used in all cases is readable ASCII text, using hexadecimal numbers for improved efficiency. However files are not directly exchangeable between microSD and PC HDD systems, you must use the MATRIX app to read/write and manage both systems. It is strongly recommended that you save backup copies of your PC files to a backup folder, in case you accidently delete or corrupt your files.

Drawing Graphics

Start by choosing the correct frame size for your project. Click on the field beneath the **Matrix** logo to do this. As LED panels are 16x16, a project using 3x6 panels has 48x96 LEDs.

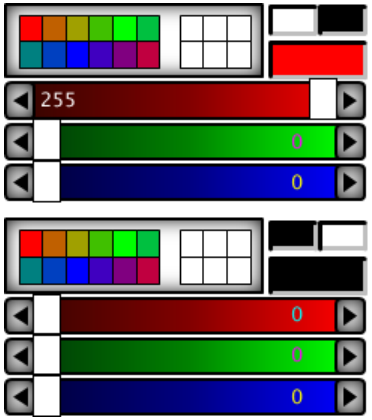
Frame sizes: 16x16, 48x16, 48x32, 48x48, 48x64, 48x80, 48x96, 48x112, 48x128



Matrix displays two windows, one for drawing in (Editor) and one used to preview files on your system. At low resolutions the Editor is above the Preview; at higher values they are side by side.

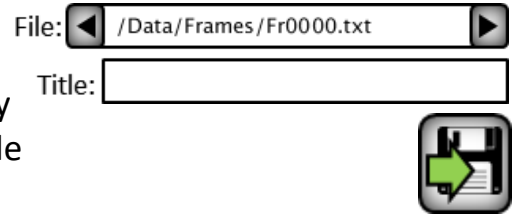
By default the freestyle pen tool is selected. Clicking on the other tool buttons changes the editors mode. The active mode is underlined in red. To draw simply click and drag with the mouse buttons. The left-hand button is the Pen colour, and the right-hand button is the background or secondary colour. As the outcome of all of the tools is dependent on the mouse buttons, it is recommended that a mouse is used, rather than a mouse pad.

A 'Zoom' feature option appears in the higher frame size settings. This causes the editor to display the image twice normal size, through a view port. Hold the SPACE key down and move the mouse to navigate around.



The ink colours can be changed by either clicking on the pre-set values or moving their respective sliders. Each colour component, Red, Green and Blue has a range of 0 – 255. When working on an image with many colours, you can easily switch to an existing colour, picking it by holding down the CTRL key whilst clicking on the pixel.

Start cloning operations by holding the CTRL key down whilst clicking on a reference point. Now drawing in another area, will effectively copy the source pixels into that new area. Pressing the ESC key ends cloning, and is used to abort all of the other drawing modes.



Select a file path using the file selector and save your image by clicking on the SAVE button. As there is only one level of UNDO, it is recommended that you save your work frequently. Giving your image a recognisable title will help you recognise it in the file list later, as all filenames are 8 character numeric.

Files & Folders

As explained in a previous slide, files are store either on your PC, or in the microSD card connected to your microcontroller, and finally on both.

The yellow ‘folder’ buttons enable you to view lists of files stored on your PC, as either Animation, Frame graphics or Sprite16 graphics files. Along with the filenames, Matrix reads the ‘Title’ information entered for each file and displays this too, in blue. You can load files by double clicking on the list item. For graphics files you need to have the correct frame size setting. Sprite16 files will only load into a frame size of a 16x16 array.

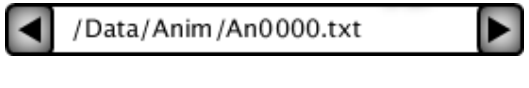
If you click and drag on filenames in either list of graphics files, they will appear in the preview window. This is also true if you click and drag in either the file path or preview path fields. Think of them as sliders.

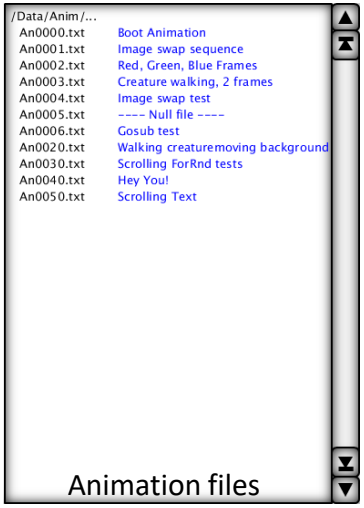
Whilst there are separate buttons for storing graphics files to either the PC HDD or microSD card, via USB; you can hold down the SHIFT key when saving graphics to your PC and it will also be transferred to the micro...

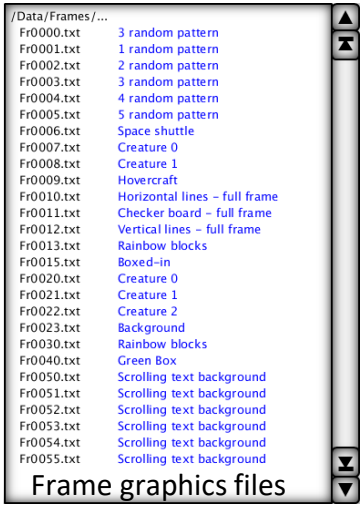
Shift + 

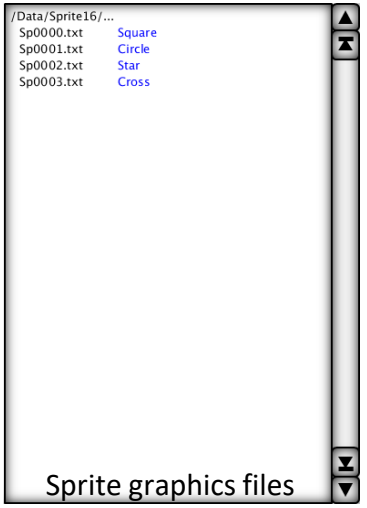
Whilst editing graphics, Matrix will send data to the micro, such that it can mimic what you are drawing on an LED display. But this is only a temporary process, and not the same as saving the file to the microSD. Save files regularly to avoid data loss.

Double clicking on a file, listed in the Animation list, will invoke the script editor mode and load that file from your PC HDD. See animation scripts described on the next page. You can also select one or more files for deletion or for sending to the microSD card using the synch buttons. 



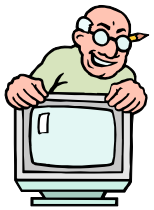






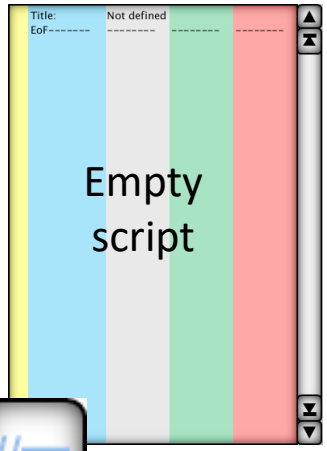
PC file lists

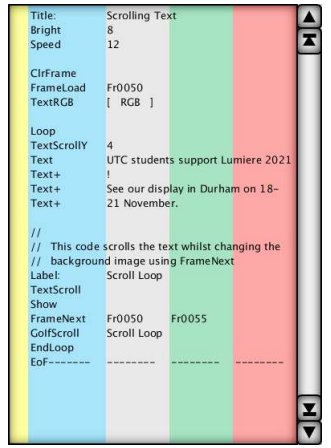
Animation Scripts



The animation script editor is accessed by clicking on the button, top right, or by double-clicking on a file in the file list. This reveals a multi-coloured list. New lines are added to the list by clicking on the New LINE button. When lines are added they are set as being selected, and are therefore highlighted.

Switch to Animation editor





To remove line selections press the ESC key at any time. The left mouse button adds lines before the last or selected line, where as the right mouse button adds a line after the first selected line. This helps when adding lines to existing scripts. To select lines click in the left-most yellow column.

The animation editor mode also reveals a new set of hidden tools, such as line DELETE, CUT, COPY and PASTE buttons. DISABLE is used in debugging.

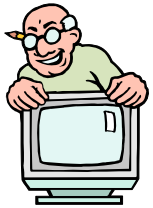
An important tool is the command reference list, which holds all of the commands that the associated micros animation engine will respond to. Placeholder commands, not yet coded are in red. Commands are added to selected lines by double clicking on items the reference list. This will add commands to new lines or overwrite existing commands, whichever is selected. And the process moves down the list of selected lines progressively.

The '**Help**' system describes each command under the mouse pointer, and the line is also highlighted, in either list. Pointing at RGB values in the script, list will change the line highlight to reflect the set colour. To change values associated with commands, simply click on them, mouse left or right. RGB values are changed using the pen colour selector. Other values are incremented or decremented using the left and right mouse buttons. Text is modified using a simple line editor; press <RET> or <ESC> keys to either accept or reject changes.

Enter a title for your script, to make the file more recognisable later, as seen in the animation file list.

The animation scripts are saved to the PC HDD using the file selector and SAVE button. They can also be transferred to the microSD card using a button at the top of the MATRIX app. That is required for standalone operation, when a PC is not connected to the ESP32 microcontroller.

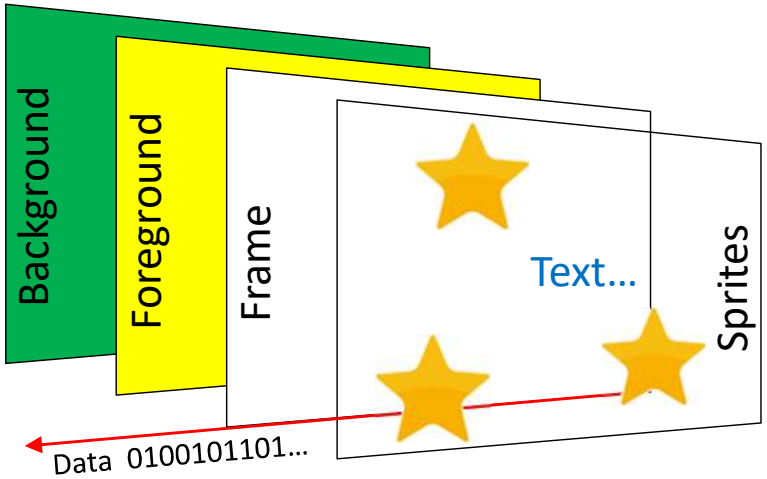
Animation Planes



Traditional animation frames are made from transparent sheets, layered one on top of another, to give the combined scene. This saves on having to redraw everything in the scene for each frame, and makes it easier to move or change each layer of the scene independently.

In a similar way the micro used to drive the LED matrix display, can be instructed to store RGB pixel information in separate arrays, like layers, and then combine them into one array before sending the data to the LED matrix when given a 'show' command.

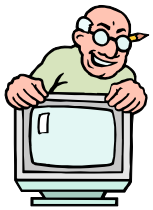
The array holding the final image is known as the 'Frame' and the two supporting arrays are the 'Foreground' and 'Background'. All three arrays can have images loaded into them, from instructions in the animation script. Once loaded the images can have functions applied, like Dim and Rotate, or be cleared before loading another image. That is often necessary, as images are often transparent (ie. pixels not set) and we don't want to leave remanence behind. The background and foreground need to be loaded into the frame, to be included in a scene, and can be loaded in either order, one effectively overlaying the other.



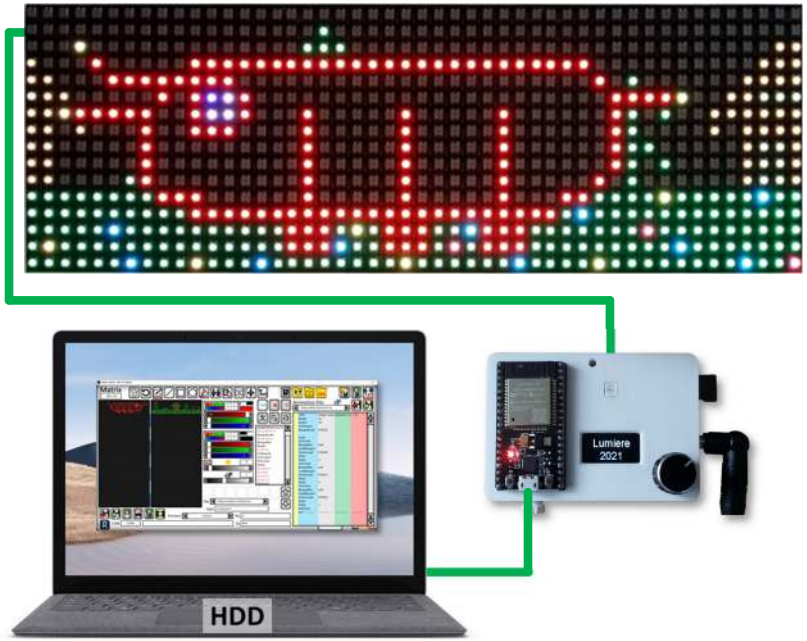
The Frame array has additional functions like the placement of characters or text, and even scrolling text. Up to eight small arrays of 16x16 LEDs, defined here as sprites, can also be positioned in the scene and switched ON or OFF. Sprites that are set to visible, will be automatically drawn in the Frame, prior to Show events. Simple animations might only use the Frame array, for example to cycle through a series of images; where as complex scenes will make use of all four planes, and incorporate sprites, scrolling or dimming effects.

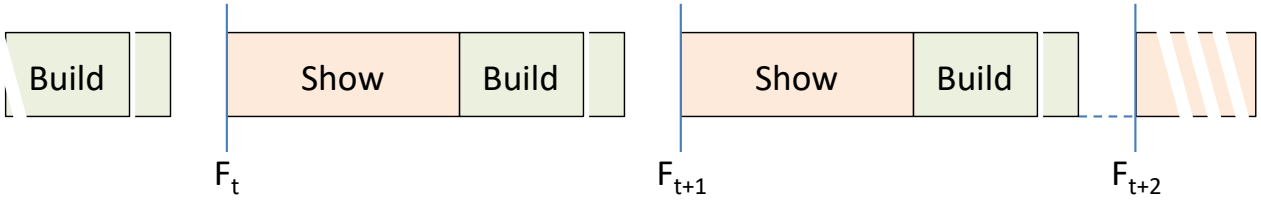
It should be known that the time needed to load an image into an array from the microSD card is proportional to its contents, as only set pixels contain data. The data from the Frame array is sent to the WS2812B LEDs at 800 kb/s, so the overall number of 16x16 LED panels used in your project will have an inverse effect on frame rate. For example a 3 panel 48x16 display can run at > 40 fps, where as a 18 panel 48x96 display will only run at 7 fps. Acceptable frame rates for most animations vary between 5 – 15 fps.

Animation Engine



The animation scripts are sent to the microcontroller, which runs code on the ESP32 known as the animation engine, AE for short. It's role is to interpret each command in the script in sequence, in order to build the output frame, and then send it to the LED panel display at regular intervals.

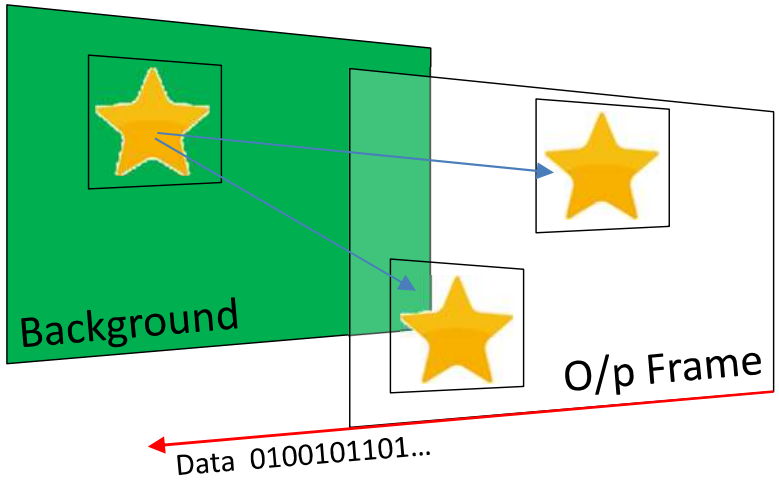




In the timing diagram above the AE is sending LED colour data to the display panel at regular intervals using the FastLED 'Show' command. The frame rate speed is set such that there is time after each burst of 'Show' data, to build the contents of the next frame. The time needed to build the frame is dependent on the animation script, and it will vary.

A healthy state is when there is always a time gap between building a frame, and the pending show event. In this condition the desired frame rate is being achieved. The 'Show' period is constant for a given panel, and dependent on the number of LEDs in your display, with RGB data being sent at a precise 800 kbps. Therefore the show period is proportional to the size of your LED panel, and the maximum frame rate is inversely proportional to the number of LEDs. So large panels limit the maximum frame achievable.

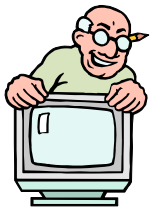
When creating scripts, which will build your animations, try to consider of the amount of data you are moving around in the process. For example we could load a background frame which contains only a small image. Whilst there are commands to copy the whole of the background into the output Frame, it is far more time efficient to only copy in the area of the small image from the background you want to display.



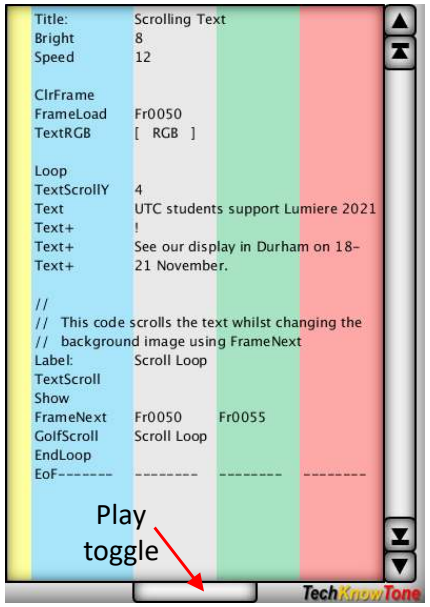
You could in fact build the output Frame by copying in a series of small images taken from the background. And these in turn could be placed in different locations, using variables to control the copy process; there by developing movement from a static background source.

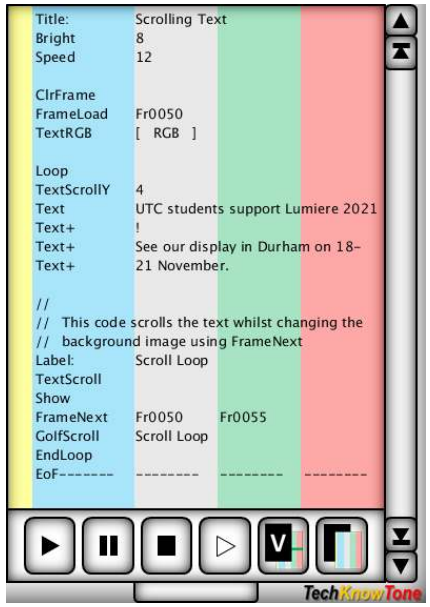
The same applies to use of the Foreground frame, for Frame construction.

Playing Animations



The animation script and graphics, created with the Matrix editor, are transferred to the micro over USB, which in turn sends serial data to the LED matrix. At the base of the script window is a thin button which, when clicked, reveals controls for playing the animations.





For animations to run, the graphics need to have been stored on the microSD card, but the script can be sent dynamically; allowing you to make changes and run them on the fly. Once you are happy with the animation you can transfer it to the microSD and run it directly by clicking on the running man icon above the list.

In this way the PC could then be disconnected and the micro would run the script independently, until reset or power is removed. The Matrix controls enable you to PLAY, PAUSE, STOP and single STEP through the script. As the micros code runs the script, it reports to the PC via a black console window what it is doing.

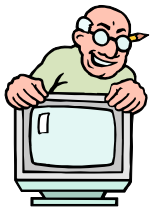
This form of verbose messaging can be reduced to simple line numbers, by clicking on the VERBOSE toggle button. Then when you return to the script view, the line highlight will move in line with the micros messaging, to show which part of the script the animation is running. Loaded images will also be previewed, as they are being loaded in the animation.

Auto-Boot

In order to initiate animations being run, from power-up or reset, without the need for a PC to be connected, the code in the micro checks to see if the file An0000.txt is present in the Root/Anim/.. folder on the microSD card. If it is, then the micro will load it into the animation engine and run the script stored in it.

The Matrix app can create a script of scheduled animation files, which the user wants to run consecutively as a whole series of short animations, one after the other. The short animations, referenced in this list, simply use an AnimSchESC command, to instruct the background scheduler to load the next animation script in the sequence.

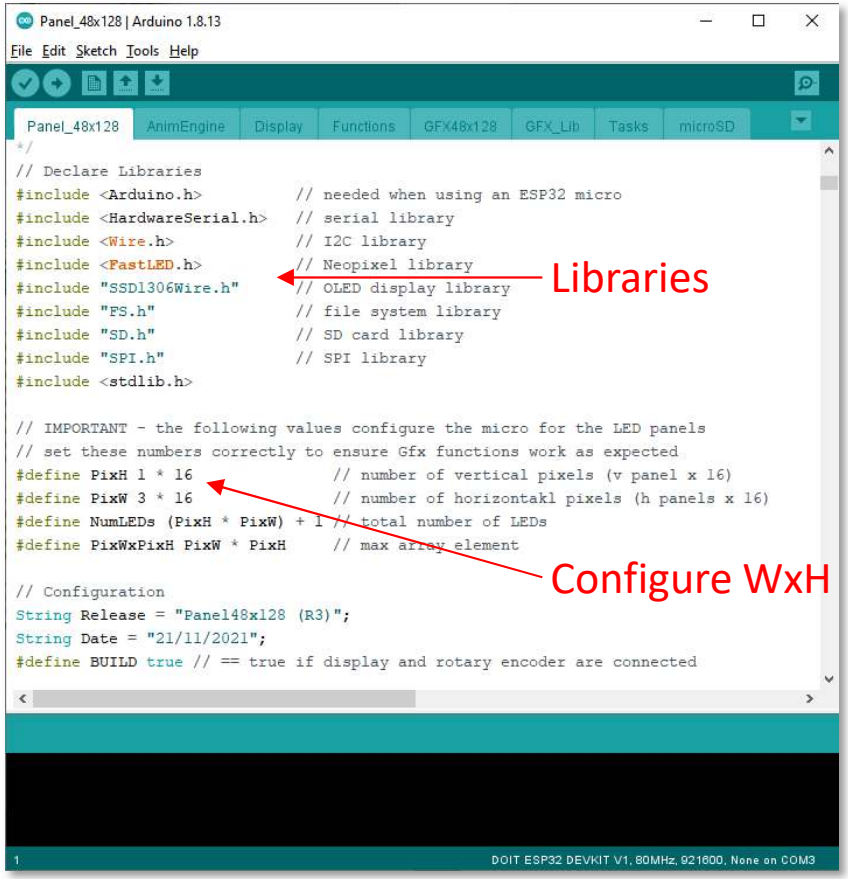
ESP32 microcontroller



The ESP32 microcontroller is an essential part of this system, as it contains the C++ code which forms the animation engine, which draws data from files held on the microSD card, to generate your light shows. This great microcontroller is similar in many ways to an Arduino, but it is much more powerful, running two 32-bit cores at 80 MHz, with a wealth of memory.

In this project the ESP32 has been programmed using the Arduino IDE, which works well, provided that you install the ESP32 board libraries and associated USB drivers. You can see the libraries used in this project in the image, along with definitions of the display.

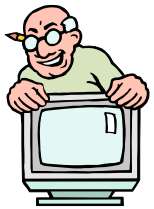
By default the code is set for a display, consisting of 3 horizontal 16x16 panels, giving a matrix of 48x16 pixels (WxH). But the Matrix app and micro can handle much larger display, up to 48x128 pixels, that's 6144 LEDs! This limit was chosen due to the fact that the technology effective slows down as the number of LEDs increase, as the data rate is fixed at 800 kbps. The default 48x16 panel can run at frame rates in excess of 40 fps (much fast than video @25 fps), but a larger 48x128 needs much more data, dropping the frame rate to 7 fps. But that is still quick enough for most animations.



The controller has a 64x128 OLED display and a rotary encoder included in the design, but you can if you wish build it without these components. If you don't wish to have a controller with a menu system, the only essential components are the ESP32, the microSD card reader and the single WS2812B LED. As explained in the wiring diagrams, the single LED acts as an electronic level-shifter, between the 3.3v micro and the 5v used by the LED panels.

The microcontroller, if instructed by an IcMax command, can determine the predicted current to be used by the LED display on the fly, and therefore limit the brightness limit used. This is a great feature, as it allows you to design your graphics, without worrying whether or not your 2A limited supply can cope with the demand.

Command Reference List



There is a wide range of commands available in the Matrix Editor, which are described in much more detail in the Command Reference document, than in the simple dynamic Help system provided by the Matrix App.

As well as loading images for frames and sprites, you can also set variables and use conditional ‘If’ statements. Variables can also be used to position images and adjust brightness and speed.

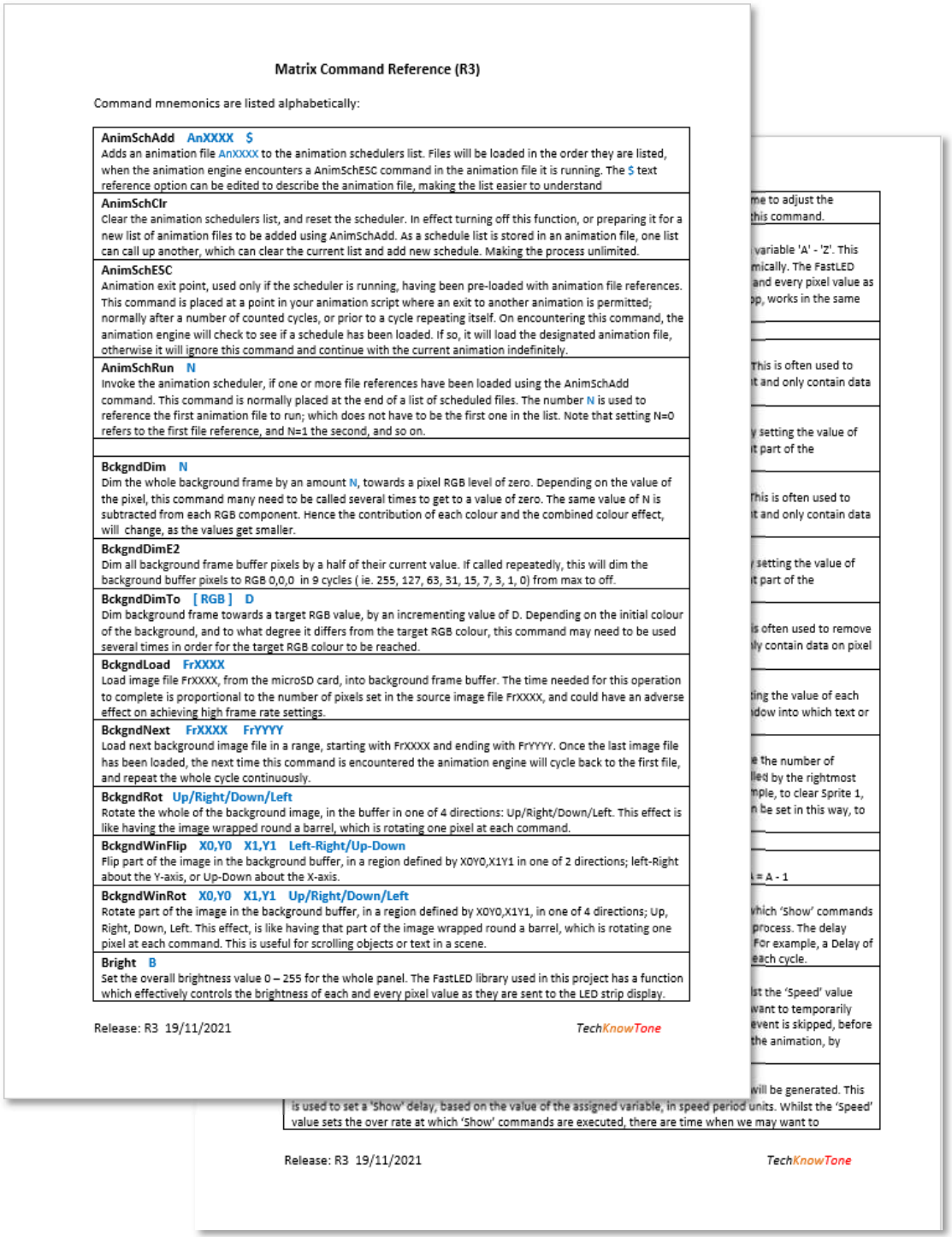
There are also commands for creating schedules, which control the order in which animation files are loaded and played.

It is well worth spending some time reading through this list, to familiarise yourself with what is available. Each command is listed alphabetically for easy reference.

The parameters associated with each command are deliberately constrained within the list editor. All you need to do is click on them with either the left or right mouse key to change their values, or to enter into special editing modes for text and RGB colours.

In order to reduce the length of a script, some of the commands use single bit assignments to apply the same parameter to all of those with a bit set in the mask. An example of this is the SpriteOn{M} command, where the 9-bit field {xxxxxxxxx} relates to all 9 sprites, so you can control all nine of them with one command.

An animation script can be up to 1,000 lines in length, so there is plenty of scope for making quite complex animations. If you wanted more, then you could easily combine them as a schedule.



Matrix Command Reference (R3)

Command mnemonics are listed alphabetically:

AnimSchAdd AnXXXX $
Adds an animation file AnXXXX to the animation schedulers list. Files will be loaded in the order they are listed, when the animation engine encounters a AnimSchESC command in the animation file it is running. The $ text reference option can be edited to describe the animation file, making the list easier to understand

AnimSchClr
Clear the animation schedulers list, and reset the scheduler. In effect turning off this function, or preparing it for a new list of animation files to be added using AnimSchAdd. As a schedule list is stored in an animation file, one list can call up another, which can clear the current list and add new schedule. Making the process unlimited.

AnimSchESC
Animation exit point, used only if the scheduler is running, having been pre-loaded with animation file references. This command is placed at a point in your animation script where an exit to another animation is permitted; normally after a number of counted cycles, or prior to a cycle repeating itself. On encountering this command, the animation engine will check to see if a schedule has been loaded. If so, it will load the designated animation file, otherwise it will ignore this command and continue with the current animation indefinitely.

AnimSchRun N
Invoke the animation scheduler, if one or more file references have been loaded using the AnimSchAdd command. This command is normally placed at the end of a list of scheduled files. The number N is used to reference the first animation file to run; which does not have to be the first one in the list. Note that setting N=0 refers to the first file reference, and N=1 the second, and so on.

BckgndDim N
Dim the whole background frame by an amount N, towards a pixel RGB level of zero. Depending on the value of the pixel, this command many need to be called several times to get to a value of zero. The same value of N is subtracted from each RGB component. Hence the contribution of each colour and the combined colour effect, will change, as the values get smaller.

BckgndDimE2
Dim all background frame buffer pixels by a half of their current value. If called repeatedly, this will dim the background buffer pixels to RGB 0,0,0 in 9 cycles (ie. 255, 127, 63, 31, 15, 7, 3, 1, 0) from max to off.

BckgndDimTo [RGB] D
Dim background frame towards a target RGB value, by an incrementing value of D. Depending on the initial colour of the background, and to what degree it differs from the target RGB colour, this command may need to be used several times in order for the target RGB colour to be reached.

BckgndLoad FrXXXX
Load image file FrXXXX, from the microSD card, into background frame buffer. The time needed for this operation to complete is proportional to the number of pixels set in the source image file FrXXXX, and could have an adverse effect on achieving high frame rate settings.

BckgndNext FrXXXX FrYYYY
Load next background image file in a range, starting with FrXXXX and ending with FrYYYY. Once the last image file has been loaded, the next time this command is encountered the animation engine will cycle back to the first file, and repeat the whole cycle continuously.

BckgndRot Up/Right/Down/Left
Rotate the whole of the background image, in the buffer in one of 4 directions: Up/Right/Down/Left. This effect is like having the image wrapped round a barrel, which is rotating one pixel at each command.

BckgndWinFlip X0,Y0 X1,Y1 Left-Right/Up-Down
Flip part of the image in the background buffer, in a region defined by X0Y0,X1Y1 in one of 2 directions; left-Right about the Y-axis, or Up-Down about the X-axis.

BckgndWinRot X0,Y0 X1,Y1 Up/Right/Down/Left
Rotate part of the image in the background buffer, in a region defined by X0Y0,X1Y1, in one of 4 directions; Up, Right, Down, Left. This effect, is like having that part of the image wrapped round a barrel, which is rotating one pixel at each command. This is useful for scrolling objects or text in a scene.

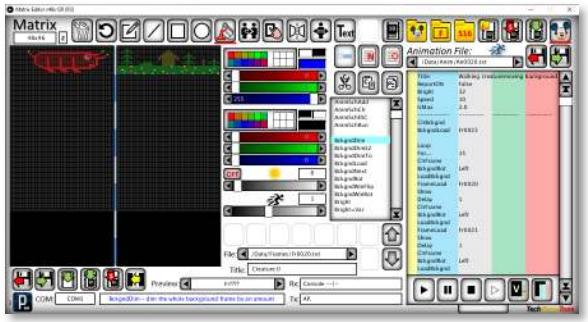
Bright B
Set the overall brightness value 0 – 255 for the whole panel. The FastLED library used in this project has a function which effectively controls the brightness of each and every pixel value as they are sent to the LED strip display.

Release: R3 19/11/2021 TechKnowTone

me to adjust the
this command.
variable 'A' - 'Z'. This
mically. The FastLED
and every pixel value as
op, works in the same
This is often used to
t and only contain data
y setting the value of
t part of the
This is often used to
t and only contain data
setting the value of
t part of the
s often used to remove
ly contain data on pixel
ting the value of each
dow into which text or
e the number of
led by the rightmost
mple, to clear Sprite 1,
n be set in this way, to
L = A - 1
which 'Show' commands
process. The delay
For example, a Delay of
each cycle.
st the 'Speed' value
want to temporarily
event is skipped, before
the animation, by
will be generated. This
is used to set a 'Show' delay, based on the value of the assigned variable, in speed period units. Whilst the 'Speed' value sets the over rate at which 'Show' commands are executed, there are time when we may want to

Release: R3 19/11/2021 TechKnowTone

Hints & Tips



Despite the aid of the built in Help system, the MATRIX app has a number of features which aren't immediately obvious, but could save you a lot of time once you are aware of them.

Esc - the keyboard Esc key helps you to exit from drawing operations, which you may have started but don't wish to complete. It also clears the line selection in lists, and cancels dialogues.

CTRL – the keyboard Ctrl key, used in conjunction with the mouse buttons, sets the start point for cloning. In the list editor you can quickly copy and paste single lines, using Ctrl+Right button to copy and Ctrl+Left to paste the chosen command line.

Shift – the keyboard Shift key, used in conjunction with the mouse buttons, doubles up on file saving actions; saving an image to the PC HDD as well as the microSD in one click. When changing values in the editor it also increase the +/- increments to +/-10, making it much easier to get to large values. The Delete line(s) button in the editor will delete the whole list, if Shift is held down whilst clicking on it.

COM – the USB COM field will make or remake a connection when clicked with the left mouse button, and disconnect the COM port when clicked with the right mouse button, freeing up the port for the IDE Serial Monitor or other Windows apps. If the MATRIX app connects to the wrong port, simply click the field again to make it select the next available port. When disconnected the text is in red ---- or –NA-.

Projects – animations are often produced as short sequences, pooled together with the use of the animation scheduler list. Simple file organisation can make things much easier to manage. For example spacing the animation file names at intervals of say 10 to 50, and naming the associated image files within those ranges. As in: An0010.txt, with Fr0010.txt, Fr0011.txt and Fr0012.txt; then An0020.txt, etc

Backup – as a project grows in size, it is relatively easy to accidently overwrite a valuable image or script file. So, to avoid losing valuable work, make a backup folder of the Data folder, and regularly copy the contents of your working folder into the backup folder.

Frame size – should be set to suite the physical size of your project, as in 48x16 pixels. Also the C++ code in the ESP32 microcontroller needs to be configured for this size too; otherwise graphics functions performed by the micro will not work correctly. How to do this is explained at the beginning of the code, setting PixH and PixW values.

Feedback on any part of this project would be most welcomed.
Enjoy!