



4D Systems

Application Note: 4D-AN-1006

Displaying an Image in 4DGL on GFX Modules

Document Date: 2nd November 2011

Document Revision: 1.0

Description

This application note is dedicated to illustrating how to display an image on a 4D GFX module using the 4DGL language. The image needs to be converted into 4D format using the Graphics Composer. In order to carry out this application, the following items are required;

- Any 4D GFX display module
- 4D Programming Cable
- micro-SD (μ SD) Memory Card
- 4DWorkshop3 IDE Software Tool
- Graphics Composer Software Tool

Application Overview

Displaying an image on a 4D screen is one of the most quintessential applications to know how to do. This application note will walk through the steps involved in decoding an image into 4D format; the procedure required to place this onto an external μ SD card; the 4DGL code required for displaying the image; and a brief insight into the various image control functions available that can manipulate the visual appearance.



Setup Procedure

Firstly, you will need to download Graphic Composer. It can be found from the 4D Systems website below:

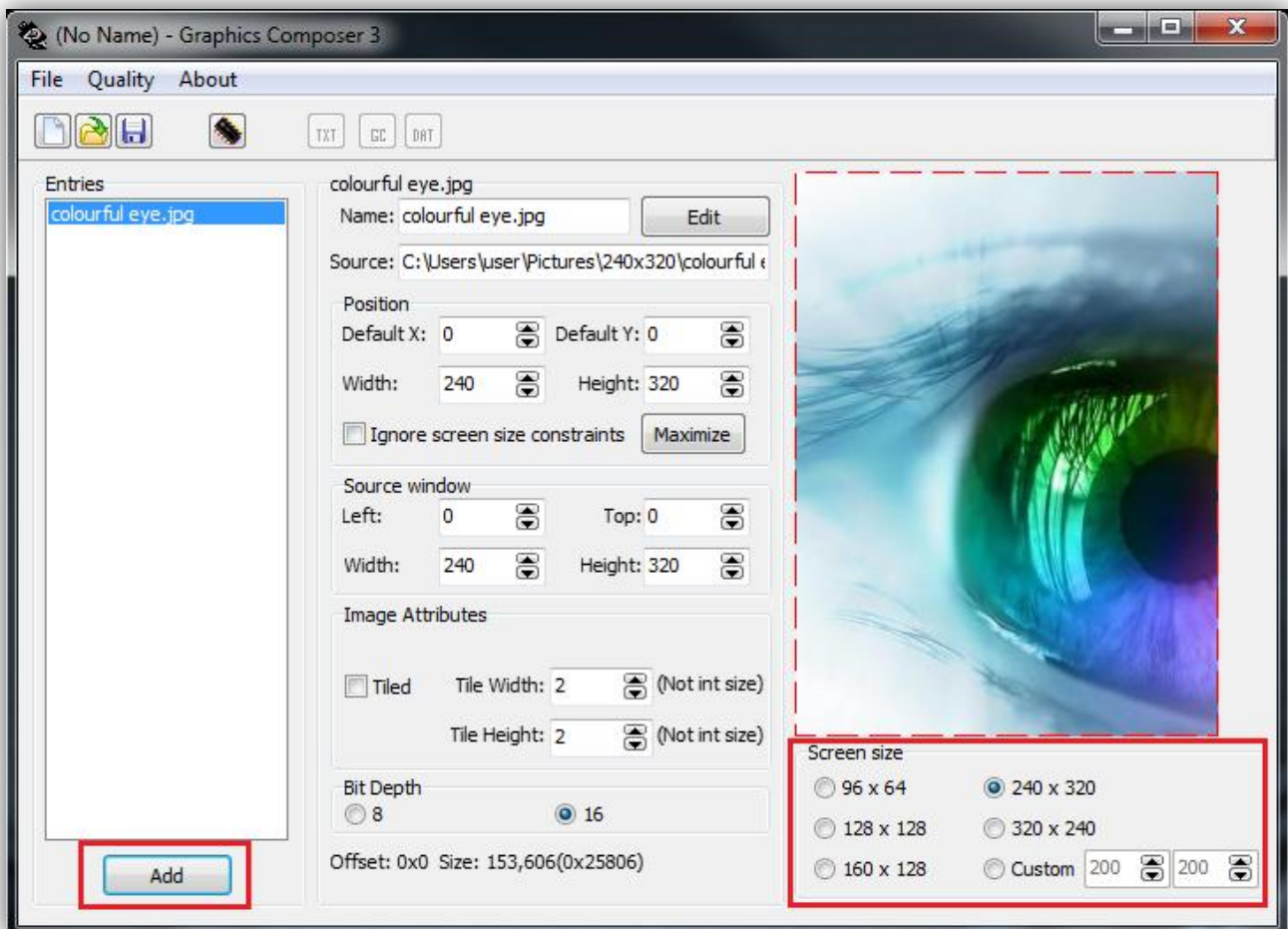
<http://www.4dsystems.com.au/prod.php?id=50>

7

Simulation Procedure

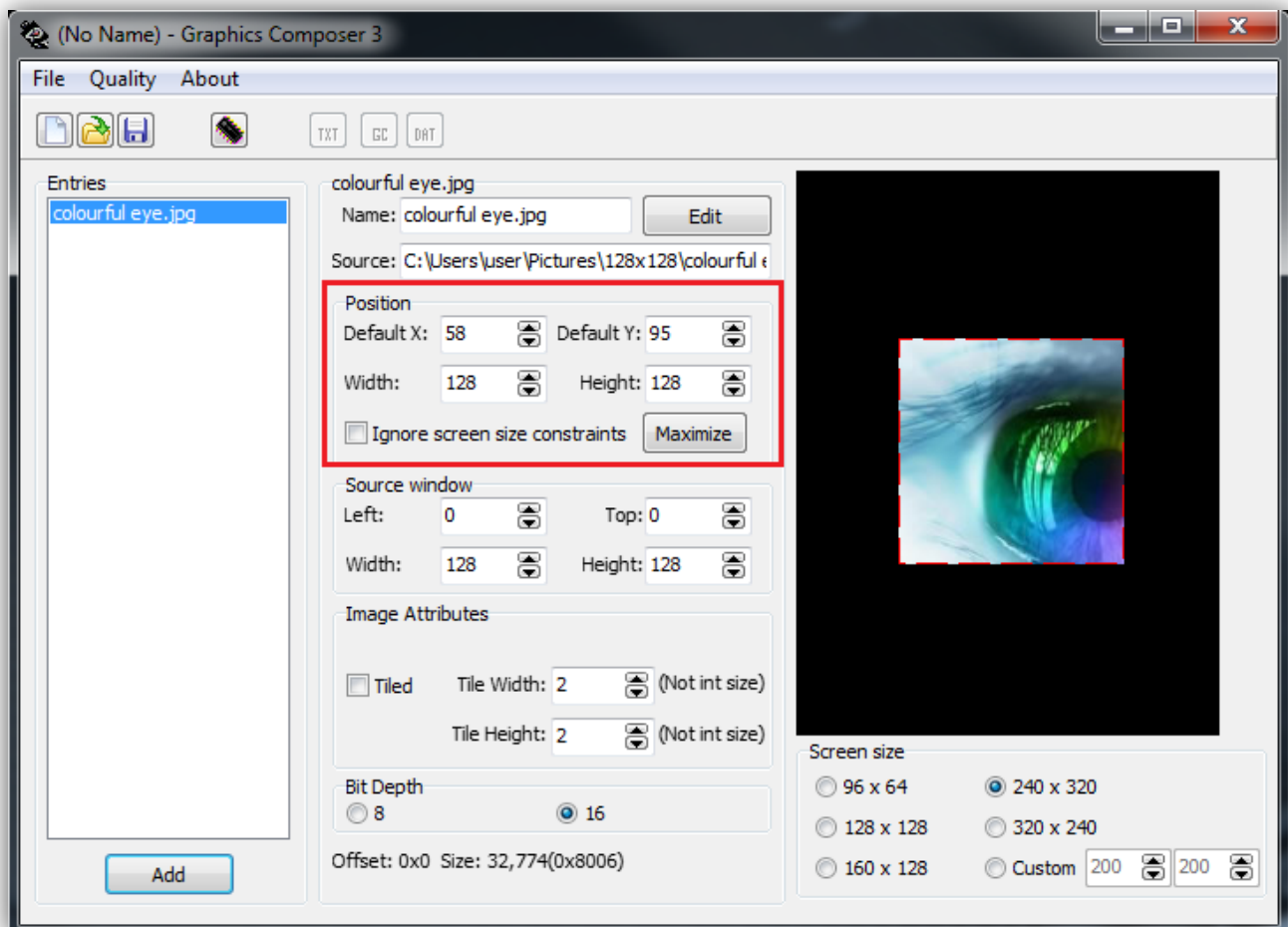
Adding an Image file

Open up the Graphics Composer and click on the **Add** button located in the bottom left hand portion of the screen. Locate the image you want to use and add it to the list, which appears in the Entries column down the left hand panel. When adding an image, the actual screen display size must be taken into consideration. For example, if using the uOLED-96-G1(GFX), the screen parameters will be 96x64, thus if a 240x320 image is added, then it will be clipped since the image is bigger than the screen. Adjust the display area size in the top right pane by selecting the appropriate **Screen Size** in the lower right hand area of the screen.



Using an Image File Smaller than the Screen Size

If an image smaller than the actual screen display area is chosen, then the user has the ability to manoeuvre the image into a desired location on the display by clicking on the image and dragging and dropping it into position. The following snapshot illustrates this point. Notice that the Default X and Y position has changed to 58 and 95 respectively. The position of the image can also be manually edited here. Remember these coordinates for programming in 4DGL.

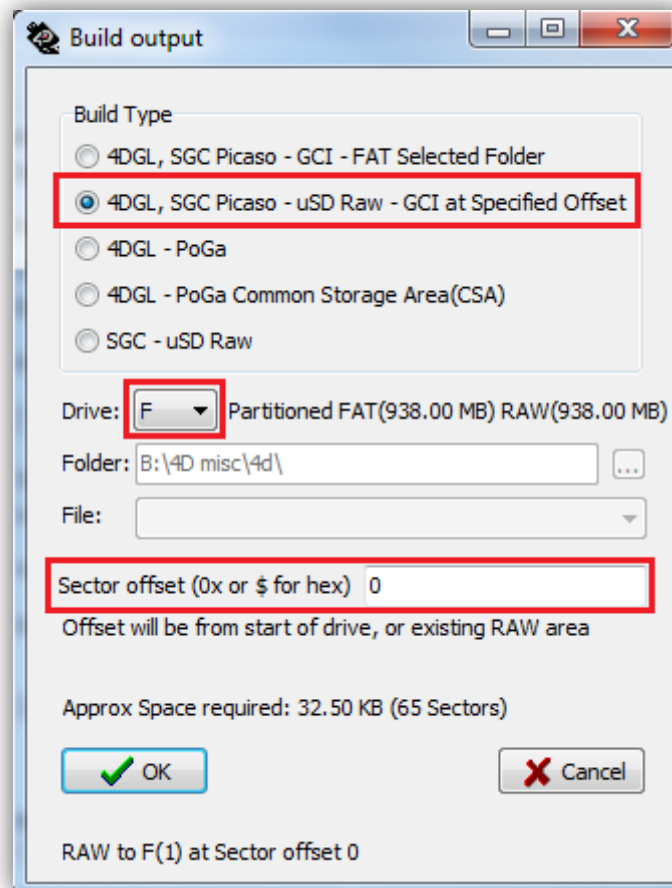


Saving the Graphics Project

Save the project as a .gcs file. Multiple images can be added to the one gcs file. This is useful as all images being used are loaded by the one file if they are required for editing in the future.

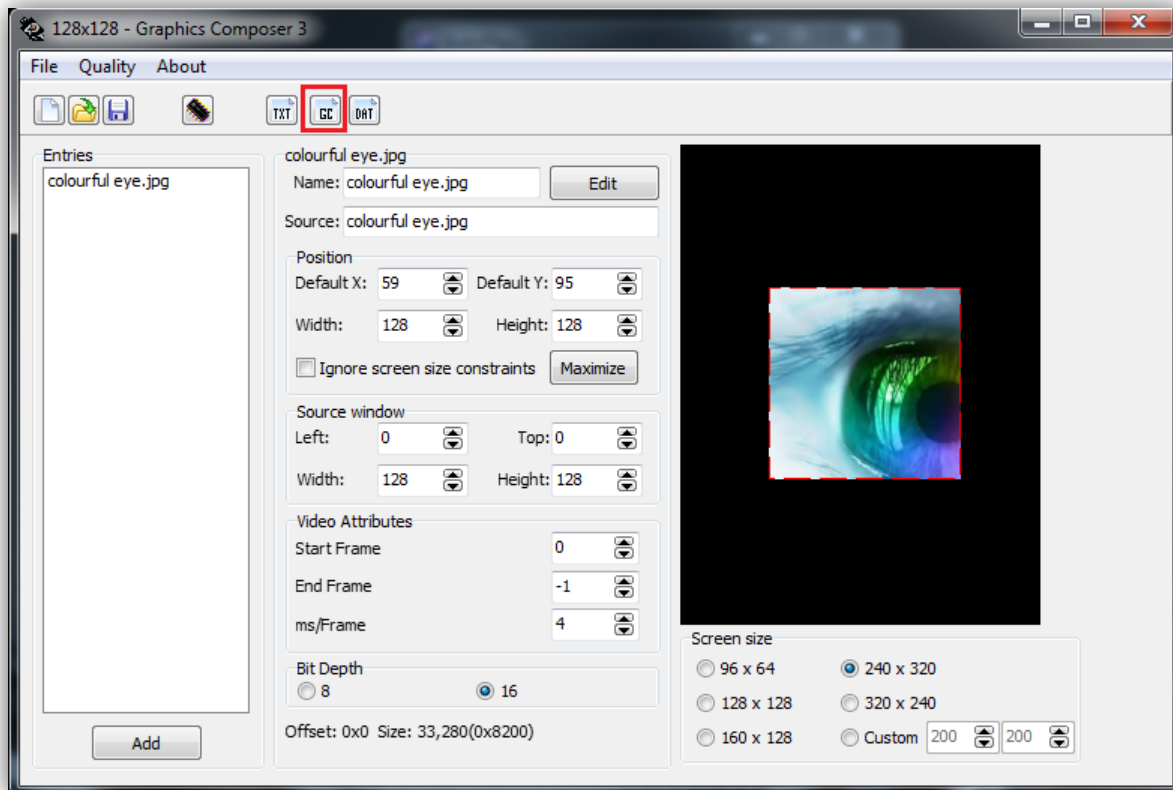
Building the Images into 4D Format

Insert a FAT formatted μ SD card into your PC and click the **Build** icon at the top of the screen as indicated by the chip. Select **4DGL, SGC Picaso – uSD RAW – GCI at Specified Offset** as the build type. Ensure that the μ SD card drive is selected and the sector offset is set to zero.



Extracting Image Sector Offsets

By clicking on the **GC** button at the top of the screen, the various sector offsets of the images can be extracted, which are necessary for use in the 4DGL code.



The following .Gc file contains the necessary data for use in the 4DGL application code.

```
128x128.Gc - Notepad
File Edit Format View Help
// Tiles - "colourful eye.jpg" size 2x2
// width = 2
// Height = 2
// Usage: colourful eye.jpg(x, y, frame);
#constant colourful eye.jpg $media_setSector(0x001D, 0x5001); media_VideoFrame
```

Example Application Code

When it comes to developing 4DGL code to display an image, there are specific commands that must be used. The following script illustrates an example of how to implement the necessary functions in 4DGL. The example uses a uOLED-32028-P1T(GFX) however, can be adapted to any GFX module, by simply changing the first line to the appropriate platform.

```
#platform "uOLED-32028-P1_GFX2"

/*****
* Filename: ImageDisplay.4dg
* Created: 2nd November 2011
* Updated 2nd November 2011
* Author: 4D team
* Description: display an image using 4DGL
*****/

#inherit "4DGL_16bitColours.fnc"

func main()

    while(!media_Init());
    media_SetSector(0x001D,0x5001);
    media_Image(58,95);

    repeat
    forever

endfunc
```

Proprietary Information

The information contained in this document is the property of 4D Systems Pty. Ltd. and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission.

4D Systems endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission. The development of 4D Systems products and services is continuous and published information may not be up to date. It is important to check the current position with 4D Systems.

All trademarks belong to their respective owners and are recognised and acknowledged.

Disclaimer of Warranties & Limitation of Liability

4D Systems makes no warranty, either expresses or implied with respect to any product, and specifically disclaims all other warranties, including, without limitation, warranties for merchantability, non-infringement and fitness for any particular purpose.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

In no event shall 4D Systems be liable to the buyer or to any third party for any indirect, incidental, special, consequential, punitive or exemplary damages (including without limitation lost profits, lost savings, or loss of business opportunity) arising out of or relating to any product or service provided or to be provided by 4D Systems, or the use or inability to use the same, even if 4D Systems has been advised of the possibility of such damages.

4D Systems products are not fault tolerant nor designed, manufactured or intended for use or resale as on line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of the product could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). 4D Systems and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

Use of 4D Systems' products and devices in 'High Risk Activities' and in any other application is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless 4D Systems from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any 4D Systems intellectual property rights.