3D Interface Specifications

White Paper

Philips 3D Solutions

Document Information

Info	Content
Title	3D Interface Specifications, White Paper
Date	15 February 2008
Security	The attached material and the information contained herein are proprietary to Philips 3D Solutions. Copying, reproduction, adaptation, modification or dissemination in whole or part is not permitted without written permission from Philips 3D Solutions.
Contact address	Philips 3D Solutions High Tech Campus 27 5656 AE Eindhoven The Netherlands, E-mail: 3DSolutions@philips.com Website: www.philips.com/3dsolutions

Table of Contents

1 Introduction

1.1 Scope and Purpose

This document contains the interface specifications of the 3D Displays as developed by Philips 3D Solutions. This document covers the physical specification as well as the data format.

All displays with type numbers 42-3D### and 20-3D### developed by 3D Solutions comply with the specification described herein.
The header described in section 6 is only supported by displays with firmware 10.7 or higher. Please check your firmware version using the Display Control Tool provided with each Philips 3D display.

The intended audiences are:

- 3D system integrators that do not make use of software from Philips 3D Solutions. The intended applications are 3D CAD design, gaming, gambling, interactive applications, etc. For these persons the sections 6.1general header location and 6.2 header description are most interesting.
- For the development of digital signage related software we refer to section 6.1 where the general header location is denoted and to section 6.3, where the standard headers are mentioned for the 3d content using the s3d, v3d or b3d extensions.
- For the development of other software, please first contact Philips 3D Solutions.

WOWvx Declipse is only supported on Displays with firmware 10.3 or higher. Please check your firmware version using the DisplayControlTool.

Philips is under no circumstances responsible for malfunctioning of third party software. Furthermore, Philips is not liable for the consequences resulting from changes of mistakes within this whitepaper.

1.2 References

The following references are only informative.

[DCT]	Display Control Tool; User manual; Philips 3D Solutions;
[DDC]	VESA Display data channel standard; Version 3; December 15, 1997;
[DDC/CI]	VESA Display data channel command interface (DDC/CI) standard; Version 1; august 14, 1998;
[DPMS]	VESA Display Power Management Signaling (DPMS) standrad; version 1.0; revision 1.0; Augustus 20, 1993;
[DVI]	Digital Visual Interface DVI; Digital Display Working Group; Revision 1.0; 02 april 1999;
[EDID]	VESA enhanced extended display identification data standard; Release A, Revision 1; February 9, 2000;
[GTF]	VESA Generalized timing formula (GTF) standard; version 1.0; revision 1.0; December 18, 1996;
[I2C]	The I2C bus specification; Version 2.1; January 2000; Philips Semiconductors;
[3DTVC]	“3D throughout the video chain” by B. Barenbrug, proceedings of ICIS 2006 (pp 366-369).

1.3 Notations

0xNN	Hexadecimal numbers are represented by using 'C' language notation.
0bNN	Binary numbers are represented by using 'C' language notation.
NN	Decimal numbers have no prefix.

2 Physical interface

The connection between the host and the 3D Display makes use of a DVI based interface, see [DVI].

Table 1. DVI specifications for the 42" displays

Parameter	Value
Resolution	1920 x 1080
Refresh rate	60 fps
Gamma	No gamma correction.
Reference white	Wx: 0.280 Wy: 0.285 (at 10500°K)
Aspect ratio	16:9
Order of sample scanning	Left to right. Top to bottom. Progressive
Coded signal colour space	RGB (8 bit coding)
Supported standards	[DDC], [DDC/CI]
Not supported standards	[GTF], [DPMS]

Table 2. DVI specifications for the 20" displays

Parameter	Value
Resolution	1600 x 1200
Refresh rate	60 fps
Gamma	No gamma correction.
Reference white	Wx: 0.313 Wy: 0.329 (at 6500°K)
Aspect ratio	4:3
Order of sample scanning	Left to right. Top to bottom. Progressive
Coded signal colour space	RGB (8 bit coding)
Supported standards	[DDC], [DDC/CI]
Not supported standards	[GTF], [DPMS]

3 Display Control Tool

The Display Control Tool is supplied with each 3D Display from Philips 3D Solutions and is running on Windows based computers. It enables the viewer to control the perceived depth and color settings real-time. It is advised to always install the Display Control Tool on each computer.

The Display Control Tool communicates with the display via the DVI cable making use of the [DDC/CI] protocol. There is no interaction on the computer between the Display Control Tool and any other software. So while developing software according to these interface specifications no consideration regarding the Display Control Tool must be taken.

The Display Control Tool requires an NVIDIA graphics card.

See [DCT] for more information about the Display Control Tool.

4 WOWvx formats

4.1 WOWvx 2D-plus-Depth format

The WOWvx 2D-plus-Depth format is very flexible. Among the advantages of using 2D and Depth images (compared to for example 9 separate views) are:

- user control over the amount of perceived depth (controlled via the header and/or I2C)
- (future) compatibility with displays which have a different number of views than the current 9, or have a different lens design which requires different interleaving (see Section 7.2), allowing the display to render in a way optimized for its optical components.
- suitability for compression for content storage and distribution
- advantages in content editing, because the disparity is explicit

More information can be found in [3DTVC]. Note: that whenever 2D and Depth is mentioned the Depth map describes the disparity and not the depth. The picture below shows how the header and the 2D and Depth sub-images are organized in a frame.

The format contains the following data:

- Header: see sections 5 and 6.
- 2D sub-image: left.
- Depth sub-image: right.
  A Depth map in grey scale picture. This depth map belongs to the 2D sub-image.

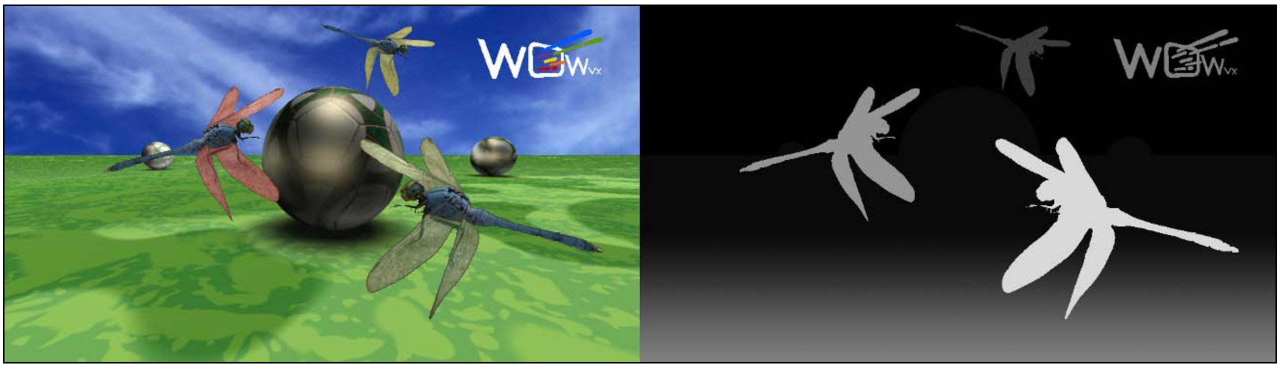


Figure 1. Overview of 3D frame in the WOWvx 2D-plus-Depth format.

The 2D and Depth sub-images are positioned besides each other, which result in a total line width equal to the native panel line width. Between each video line a blank line is added. This doubles the vertical resolution to equal the native panel vertical resolution. Sections 4.3 and 4.4 explain the 2D and Depth sub-images in more detail.

4.2 WOWvx Declipse format

The WOWvx Declipse format is an extension on the existing 2D-plus-Depth format described in section 4.1. Information on the background is added enabling the rendering algorithm filling in the occluded areas created by the foreground object. Figure 2 shows the four quadrants that should be supplied to the screen.

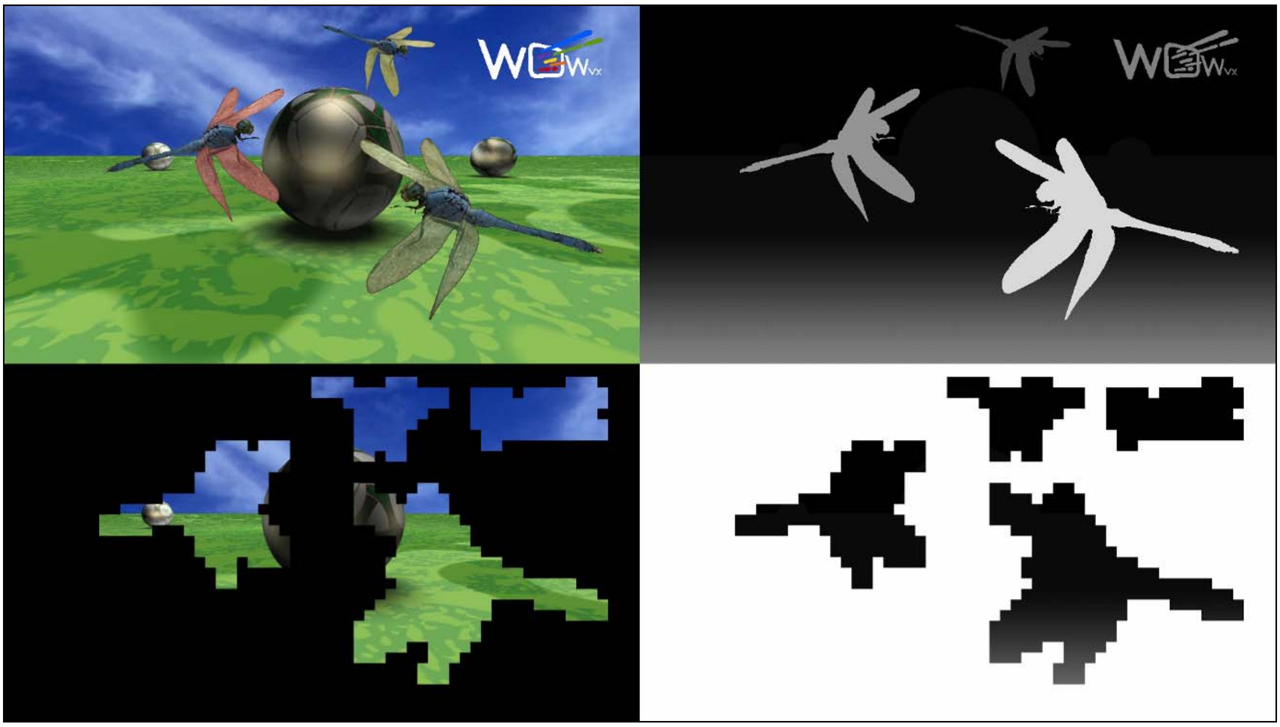


Figure 2. Overview of 3D frame in the WOWvx Declipse format.

The format contains the following data:

- Header: see sections 5 and 6.
- 2D sub-image: top left.
- Depth sub-image: top right.
  A Depth map in grey scale picture. This depth map belongs to the top left 2D sub-image.
- 2D background sub-image: bottom left.
  2D sub-image of the background. At the places where the 2D information is equal to the quadrant above (top left) the 2D information is set to black to avoid encoding redundant data.
- Depth background sub-image: bottom right.
  The Depth map on the areas occluded by the foreground objects, this depth map relates to the bottom left 2D image At the places where the disparity is equal to the quadrant above (top right) the disparity is set to maximal (white) to avoid encoding redundant data.

Note that the removal of the redundant data in the 2D and Depth background sub-images is not strictly required. It is also feasible to have the background information fully available in the frame. This is especially useful for uncompressed data (e.g. applications that generate content). In the case of content that is being stored or transmitted in compressed form it is more favorable to remove the redundant data in order to improve the coding efficiency. See section 6.2.9 for more information on the two different Declipse modes.

When the Declipse format in which the redundant data is removed is used (as depicted in Figure 2), it is required that at least 6 pixels at the edge of the non-redundant data part (in the bottom layer) have to be present in every direction as in the foreground (top) layer; i.e. at least 6 pixels of redundant data in every direction around background data has to be available in the 2D and Depth background sub-images. Furthermore, it is required that the resulting background data (after having the 6 pixel overlap with the top layer) is rounded up to macroblock boundaries. So, the overlap in both layers may vary between 6 and 21 pixels (given a macroblock size of 16x16 pixels).

4.3 2D sub-images

A 2D sub-image has a resolution of half the native panel resolution in both horizontal and vertical direction. It is an R, G and B image with 8 bits per sub-pixel. No gamma correction is performed in the display. In Figure 3 an example of a 2D sub-image is given.

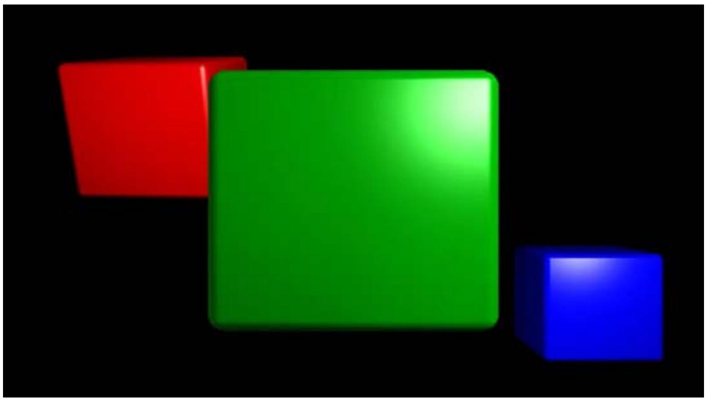


Figure 3. 2D sub-image example

4.4 Depth sub-images

A Depth sub-image has a resolution of half the native panel resolution in both horizontal and vertical direction. It contains disparity values with a range of 0 to 255, where a value of 0 corresponds with objects located with a maximum disparity behind the screen and 255 corresponds with objects located closest to the observer.

An implementation is that the Depth image is a black and white image. This means that the R, G and B sub-pixels have the same value per pixel. However the display only uses the red sub-pixels. The green and blue sub-pixels are discarded.

Note that a Depth sub-image actually contains disparity values.

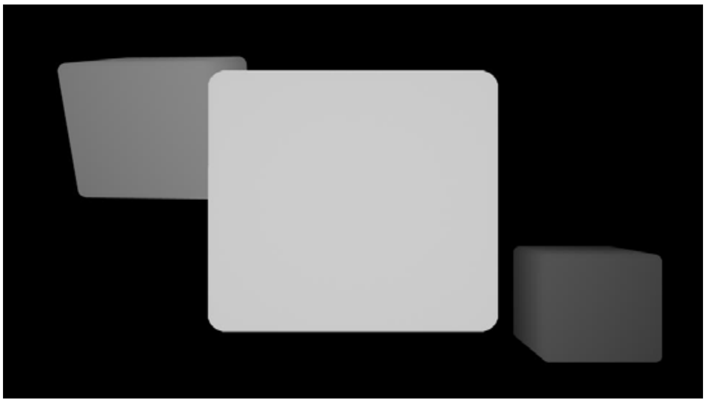


Figure 4. Depth sub-image example

5 Video data interface

The resolution of a frame that is sent to the display via the DVI cable has a resolution of 1920x1080 for 42" or 1600x1200 for 20". A frame contains the following data:

- Header
- 2D sub-image with a resolution 960x540 for 42" or 800x600 for 20"
- Depth sub-image with a resolution 960x540 for 42" or 800x600 for 20"
- Background 2D sub-image with a resolution 960x540 for 42" or 800x600 for 20"
- Background Depth sub-image with a resolution 960x540 for 42" or 800x600 for 20"

The latter two sub-images contain data in the case of the WOWvx Declipse format, while they are blank in the case of the WOWvx 2D-plus-Depth format. Section 4 explains the 2D and Depth sub-images in more detail. From now on we continue only with the explanation of 42" (1920x540=2 images of 960x540). The reader interested in 20" display has to read 1600x600= 2 images of 800x600. The rest stays the same.

More information can be found in [3DTVC]. Note: that whenever Depth is mentioned the Depth (Z) map describes the disparity and not the depth. Figure 5 shows how the header and the 2D and Depth sub-images are organized in a frame of 1920x1080 pixels in case of the WOWvx Declipse format.[1]

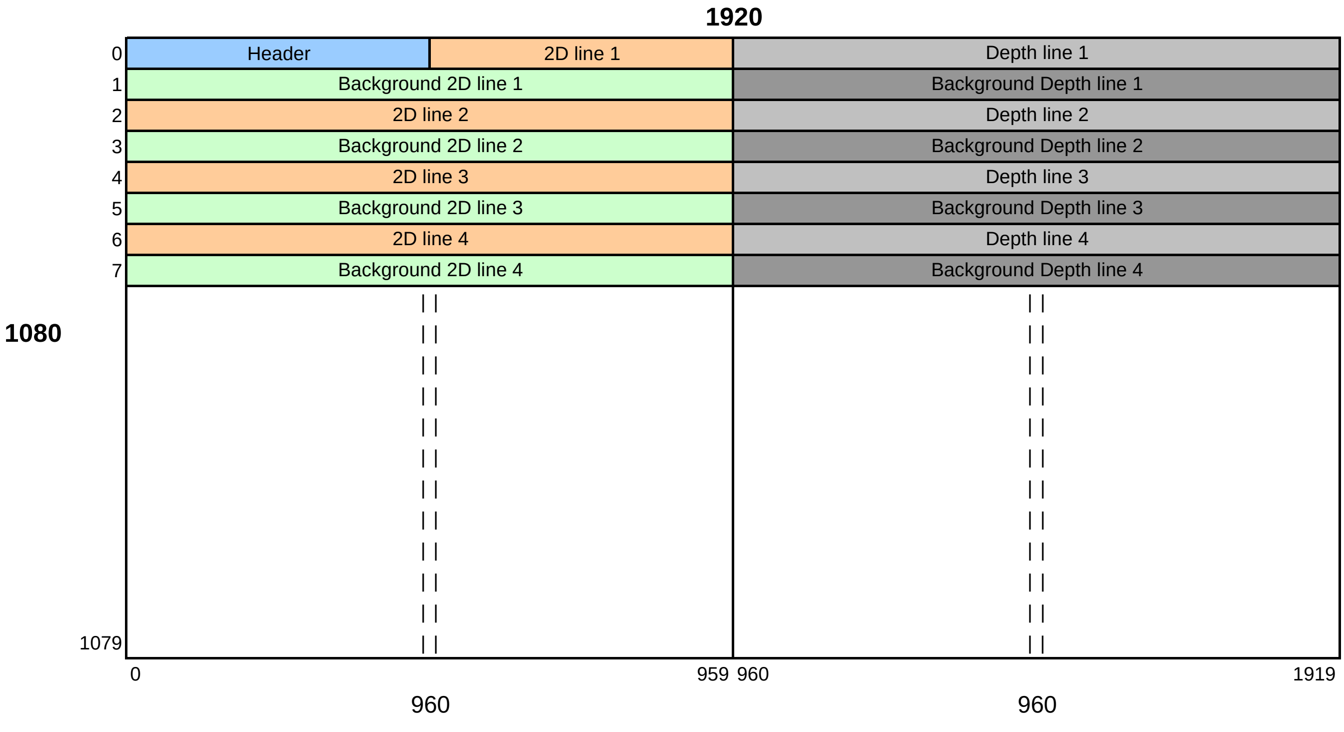


Figure 5. 3D frame layout for the WOWvx Declipse format.

The header is located in the upper left corner; i.e. the first part of the first video data line. The function of the header is twofold. When the display detects the header it switches to 3D mode and the header contains settings for rendering processing. The header is further explained in section 6.
Furthermore, a 3D frame contains 2D and Depth sub-images, both with a resolution of half the native panel resolution in both horizontal and vertical direction. The 2D sub-images are positioned in the left half of the

[1] The video data interface format towards the Philips 3D displays is different from the storage format (cf. section 4). The main reason to store content in a format with four spatially separated quadrants is to have a higher coding efficiency.

frame, and Depth sub-images in the right. The 2D and Depth sub-images of 960x540 are positioned besides each other, which result in a total line width of 1920 pixels.
For the WOWvx Declipse format the 2D and Depth sub-images of the top and bottom (background data) half of the format are line interleaved; i.e. every second line is filled with the corresponding data from the bottom half of the format.
For the WOWvx 2D-plus-Depth format a blank (or rather 'don't care') line is inserted below each line of the 2D and Depth. This doubles the vertical resolution from 540 to 1080. This is illustrated in Figure 6.

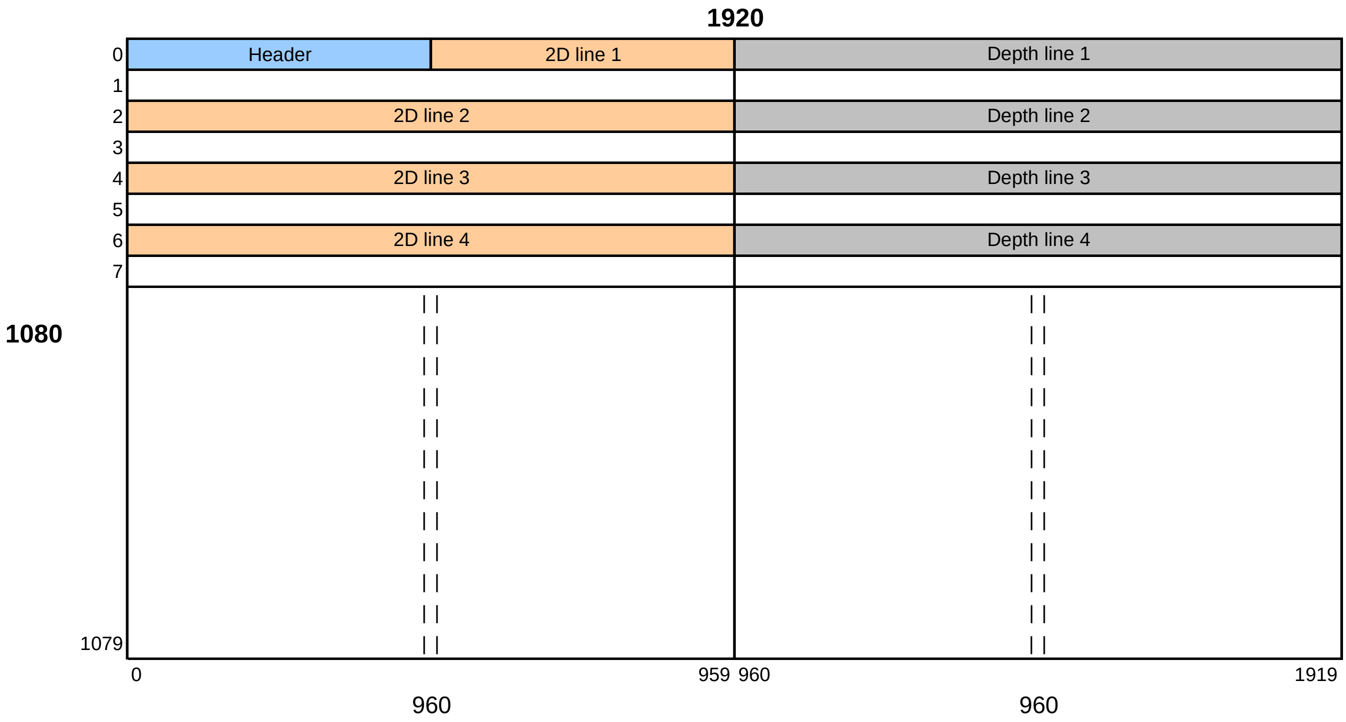


Figure 6. 3D frame layout for the WOWvx 2D-plus-Depth format.

To illustrate the difference between the storage format and the video data interface format, some concrete examples are given in Figure 7 for the different WOWvx formats. See section 6.2.9 for more information on the two different Declipse modes.

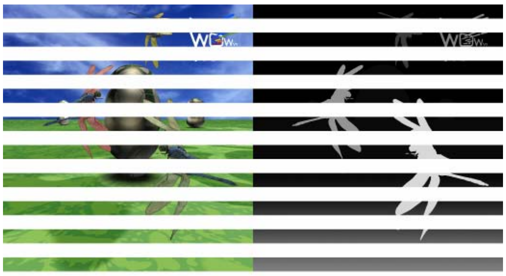
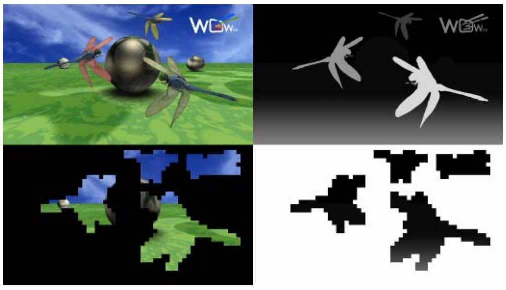
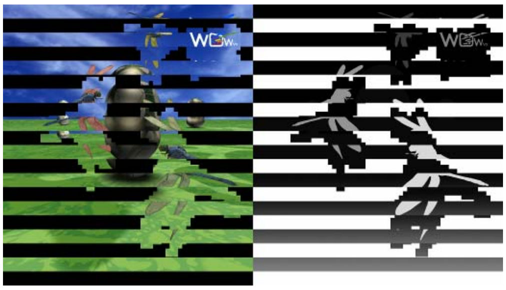
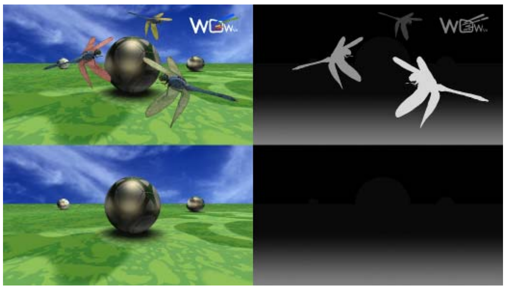
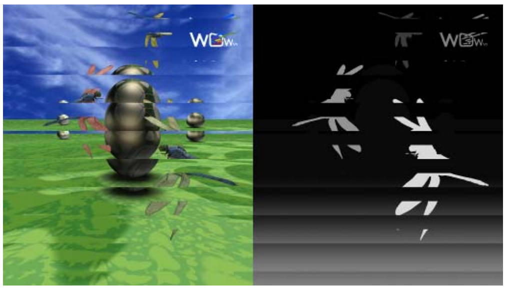
WOWvx format	Content format	Video data interface format
2D-plus-Depth		
Declipse – removed redundant data		
Declipse – full background data		

Figure 7. Difference between the storage format and the video data interface format.

6 Header

The header described in this section is only supported by displays with firmware 10.7 or higher. Please check your firmware version using the Display Control Tool provided with each Philips 3D display.
The header is positioned in the upper left corner of a frame. Each frame will contain a header otherwise the display will interpret it as a 2D frame. The header instructs the display which viewing experience is required.

6.1 Header location in the 3D frame

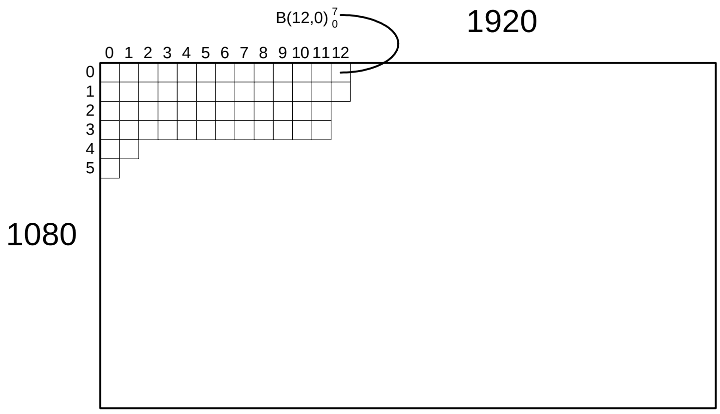


Figure 8. Numbering of rows and columns of screen.

The figure above shows how the columns and rows are numbered. Numbering starts with zero in the upper left corner. Each square pixel consists out of an R, G and B sub-pixel.
$B(12,0)_0^7$ Denotes the blue byte in the upper row in the 13th column and bits 7 down to 0. With 7 the Most Significant Bit.

The header is located in the blue sub-pixels. It is required to give all the 8 bits in the blue sub-pixel the value of the header bit. In this way the system becomes more tolerant to changes in brightness, contrast settings and noise. The header starts at the first pixel in the first row ($B(0,0)$) and onwards ($B(x,0)$). Only in the even blue sub-pixels the header is located, starting with 0, 2, 4 etc.

During rendering, the blue sub-pixels that contain the header are replaced by the blue component of their neighboring pixels. This masks the header entirely, while it hardly affects image quality.

The following formula shows which bit of which sub-pixel byte forms which bit (y) and byte (x) of the header ($x \in [0,30]$, $y \in [0,7]$):

$$B(2\cdot(7-y)+16\cdot x\,,0)^7 \;=\; H(x)^y \qquad [1]$$

The following translation table shows how the header is composed from bits in the blue sub-pixels. The first header byte, $H(0)$, is composed by combining MSB bits of blue sub-pixels.

15 February 2008

Table 3. Translation table of blue sub-pixel MSB bits into header bytes.

Sub-pixel bit	Header bit
$B(0,0)^7$	$H(0)^7$
$B(2,0)^7$	$H(0)^6$
$B(4,0)^7$	$H(0)^5$
$B(6,0)^7$	$H(0)^4$
$B(8,0)^7$	$H(0)^3$
$B(10,0)^7$	$H(0)^2$
$B(12,0)^7$	$H(0)^1$
$B(14,0)^7$	$H(0)^0$
$B(16,0)^7$	$H(1)^7$
$B(18,0)^7$	$H(1)^6$
$B(20,0)^7$	$H(1)^5$
$B(22,0)^7$	$H(1)^4$
$B(24,0)^7$	$H(1)^3$
$B(26,0)^7$	$H(1)^2$
$B(28,0)^7$	$H(1)^1$
$B(30,0)^7$	$H(1)^0$
$B(32,0)^7$	$H(2)^7$
$B(34,0)^7$	$H(2)^6$
$B(36,0)^7$	$H(2)^5$
etc.	etc.

6.2 Header data

The header is 32 bytes long and consists of two main parts: the first part of 10 bytes and the second part of 22 bytes. Declipse formats can only be used by applying both header parts. By only using the first header part (10 bytes) the content is interpreted as being in the 2D-plus-Depth format.
The table below lists the content of the 32 header bytes. The right column contains the actual value or the parameter, which is further defined in the following sections.

Byte	Content
$H(0)$	Header_ID1
$H(1)$	Hdr_Content_type
$H(2)$	Hdr_Factor
$H(3)$	Hdr_Offset_CC
$H(4)_6^7$	Hdr_Select_1
$H(4)_0^5, H(5)$	Reserved
$H(6)-H(9)$	EDC_1
$H(10)$	Header_ID2
$H(11)-H(12)$	Data_types
$H(13)-H(27)$	Reserved / Fixed
$H(28)-H(31)$	EDC_2

Unused and reserved bits **must** be set to zero.

The display interprets the header each frame, 60 times per second. Changed header values are effectuated directly.

6.2.1 Header_ID1

Indicates the format of the remainder of the header see Table 4.

Table 4. Header_IDs

Byte	Content	Description	Value
$H(0)$	Header_ID1	This header ID indicates the generic header type with a length of 10 bytes as described in 6.2.	11110001

6.2.2 Hdr_Content_type

This value defines the kind of content. Based on this the various visualization parameters, like factor and offset, are chosen that influence the rendering process.

Table 5. Content type

Byte	Content	Type value (7:0)	Type of Content
$H(1)$	Hdr_Content_type	00000000	No depth
		00000001	Signage

		00000010	Movie
		00000011	Game
		00000100	CGI
		00000101	Still
		00000110	Reserved
		etc.	

6.2.3 Hdr_Factor

Table 6. Header factor value

Byte	Content	Description	Default	Range
$H(2)$	Factor	Percentage of the display recommended depth value. (Factor/64)	64	[0-255]

Each 3D Display has a 'Display recommended depth value', which corresponds to an acceptable maximum depth factor value for that specific type of display. This value strongly depends on the lens design. The factor field in the header contains the percentage to be used from the display recommended depth value. The value of 64 corresponds with the 100% of the display recommended depth value. It is allowed to use values higher than 64. The factor works on a linear scale and is multiplied with the factor controlled by the user in the Display Control Tool.

6.2.4 Hdr_Offset_CC

Table 7. Header offset value

Byte	Content	Description	Default	Range
$H(3)$	Offset	Amount of range behind the screen. 0: Range is shifted in the direction of the viewer. 128: Range is equally divided in front and behind the screen. 255: Range is shifted away from the viewer.	128	[0,255]

Values in the Depth map equal to the header-offset value will be located on the plane of the display. All values in the disparity map with a higher value will de displayed in front of the display.

Offset_CC is the offset controlled by the Content Creator. In the system there is also an Offset_user present, which is controlled by the user using the Display Control Tool.

6.2.5 Hdr_Select_1

Table 8. Header select

Byte	Content	Description
$H(4)^7$	Hdr_Factor_select	'1': Hdr_Factor is used; '0': Hdr_content_type is used;
$H(4)^6$	Hdr_Offset_CC_select	'1': Hdr_Offset_CC is used; '0': Hdr_content_type is used;

When all select signals are low the rendering settings are set to optimal settings for the content type denoted by Hdr_content_type. By making select signals high the settings for Factor and Offset_cc can be controlled individually by the header.

6.2.6 Reserved

The bits $H(4)_0^5, H(5)$ are reserved for future use and should be set to 0 to maintain compatibility with future systems.

6.2.7 EDC_I

The 4-byte EDC field $H(6) - H(9)$ contains an Error Detection Code computed over the first 6 header bytes. This EDC uses the standard CRC-32 polynomial as defined in IEEE 802.3 and ITU-T V.42. The initial value and final XOR value are both 0. An implementation of this algorithm can be found in Appendix A:.

6.2.8 Header_ID2

Indicates the format of the remainder of the header see table 4.

Table 9. Header_IDs

Byte	Content	Description	Value
$H(10)$	Header_ID2	This header_ID indicates the second WOWvx header type with a length of 22 bytes as described in 6.2.	11110010

6.2.9 Data_types

One image can contain up to 4 sub-images. Data_types describes the contents of the 4 sub-pictures as described in Table 10. The sub-images are named A, B, C and D, Figure 9 shows how the images A, B, C and D are arranged on the interface.

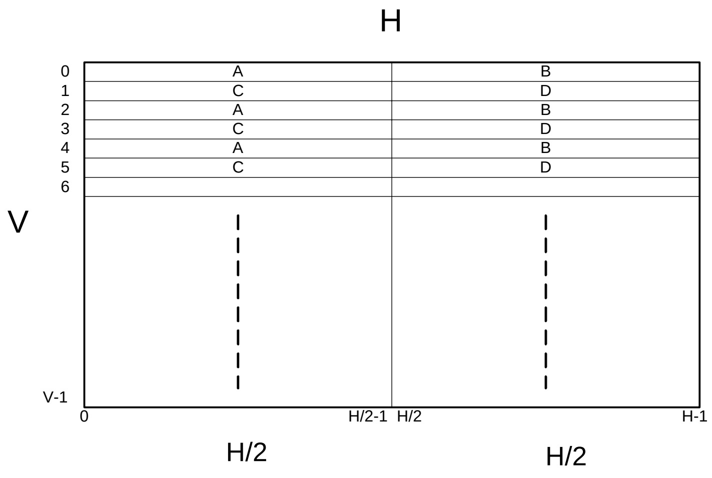


Figure 9. Naming order of the 4 sub-images.

15 February 2008

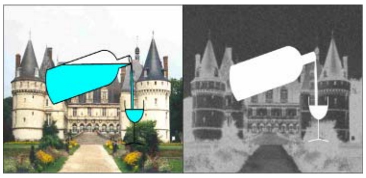
Table 10. Data type of the supporting render methods

Byte	Content	Description	Value
$H(11)$ $H(12)$	Data_type	'Traditional' 2D-plus-Depth format as denoted below	00010100 00000000
$H(11)$ $H(12)$	Data_type	Declipse – 'Removed redundant data' format as denoted below	00010100 10011010
$H(11)$ $H(12)$	Data_type	Declipse – 'Full background data' format as denoted below	00010100 11101111

Using the above information we show three supported image formats. The images are displayed as present in the video files. On the interface towards the 3D display they are interleaved as shown in figure 9 (Location).

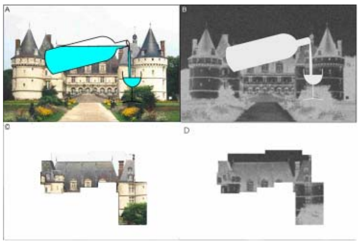
'Traditional' 2D-plus-Depth
Data_type = "00010100 00000000"

Location	Description	Reference to image
A	2D	left
B	Depth	right
C	Empty	-
D	Empty	-



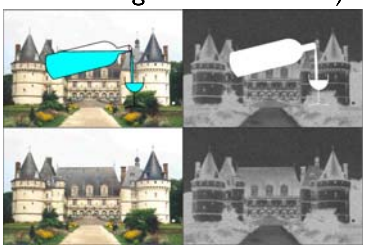
Declipse – 'Removed redundant data'
This is de default version especially designed for content that is being stored in compressed form.
Data_type = "00010100 10011010"

Location	Description	Reference to image
A	2D	top-left
B	Depth	top-right
C	2D background – removed redundant data	bottom left
D	Depth background – removed redundant data	bottom-right



Declipse – 'Full background data'
The following is specifically used for uncompressed data (as in applications that generate content).
Data_type = "00010100 11101111"

Location	Description	Reference to image
A	2D	top-left
B	Depth	top-right
C	2D background	bottom left
D	Depth background	bottom-right



6.2.10 Fixed H(13) – H(27)

These bits are reserved for future use and should be set to 0 to maintain compatibility with future systems.

6.2.11 EDC_2

The 4-byte EDC field $H(28)-H(31)$ contains an Error Detection Code computed over the first 18 bytes of the second header. This EDC uses the standard CRC-32 polynomial as defined in IEEE 802.3 and ITU-T V.42. The initial value and final XOR value are both 0. An implementation of this algorithm can be found in appendix A.

6.3 Standard headers

In many applications a standard header is sufficient. With a proper combination of both header parts a choice can be made for the content and data types (cf. sections 6.2.2 and 6.2.9). For example, the 3DS Media Player inserts a header depending on the content type derived from the file extension (*.s3d, *.v3d or *.b3d) and the data type (Declipse or 2D-plus-Depth). Standard headers for both header parts are denoted in Table 11 and Table 12. Note that for the use of the 2D-plus-Depth format the use of the first header part is sufficient, but both header parts can be used as well. For the use of Declipse content, however, both header parts are required.

Table 11. Standard headers – first header part

Header 1	Content type (section 6.2.2)	Extension
F10140800000C42DD3AF	Used for signage content	s3d
F102408000001F3A7B38	Used for video content	v3d
F10540800000E4D9502C	Used for stills	b3d

Table 12. Standard headers – second header part

Header 2	Data type (section 6.2.9)
F21400000000000000000000000000000036958221	2D-plus-Depth
F2149A000000000000000000000000000006BF6C689	Declipse – removed redundant data
F214EF00000000000000000000000000002FF0C45F	Declipse – full background data

Section 6.1 contains more information on the location of the header and how to code it into the image.

7 Operation modes

The display has two operational modes, i.e. 2D mode and 3D mode, which are described in sections 7.1 and 7.2 respectively.

7.1 2D Mode

At the moment that the display detects no header it switches back to 2D mode. The 2D mode is achieved by placing the 1920x1080 pixels on the display after a image processing step that lowers the visibility of the optical coupling between the LCD and the lens layer. Thereby a high quality 2D image is visualized.

7.2 3D Mode

The image processing within the display takes place in 3 steps, see Figure 10.
First a frame with a header and a 2D and Depth sub-image is applied to the DVI input connector. A demultiplexing block decomposes the 3D frame into the header and the 2D and Depth sub-images. These 3 components are applied to the rendering block.
The rendering block generates 9 images, which have all a slightly different camera positions. The amount of perceived depth and other depth related parameters are controlled by the values in the header.
The 9 different images are fed to the interweaving block. This block ensures that each sub-pixel is exactly located under the right lens, which ensures the best 3D experience. The interweaving process is optimized for the optical behavior of the lens layer.

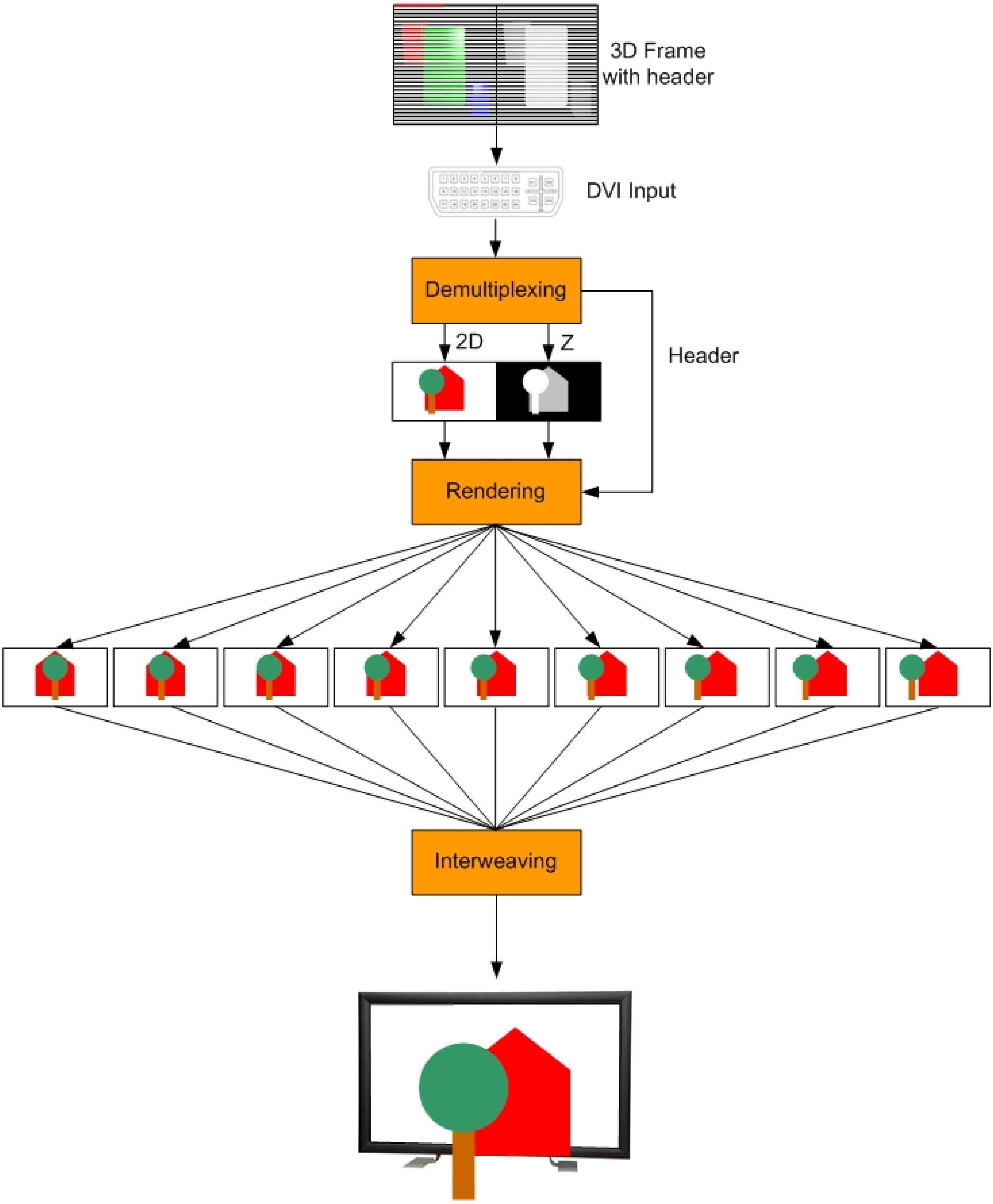


Figure 10. Overview of 3D image processing in the display.

8 Stretching of images

If an object positioned at the left edge of the image has a depth that puts it behind the screen, a viewer looking at it from the right would expect to see more pixels to the left of the object where the frame of the display would no longer be concealing the object. Because these pixels are not in the original image, the image is stretched slightly before rendering to prevent this problem. Figure 11 shows how the image is stretched. Stretching and rendering takes place inside the display.

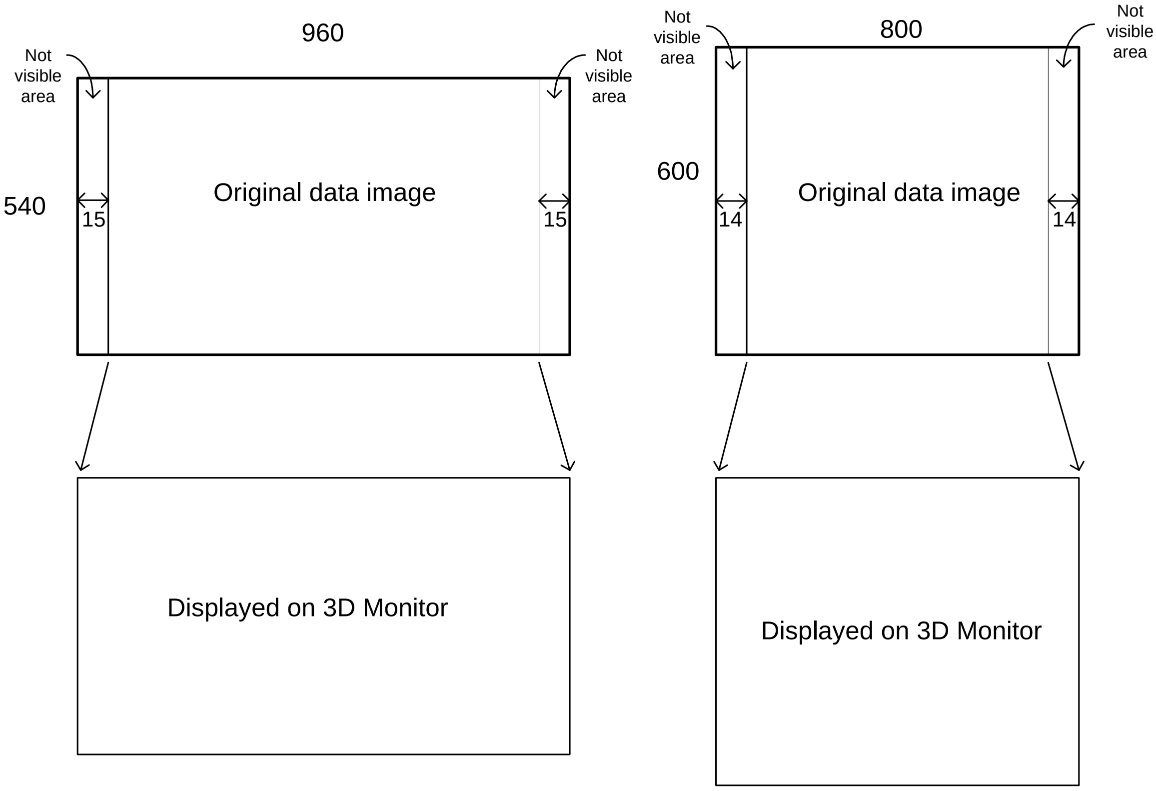


Figure 11. A band at the left and right side of the image data are not displayed on the 3D Display.

The original 2D input data has a resolution of 960x540. The leftmost pixels and the rightmost pixels of each row are not visible on the display, when the depth of the pixels is on the screen. However when the depth is behind the screen, the observer is able to see this information in these bands.
So while sending content to the display the designer must keep in mind that small bands at the left and right side will mostly not be visible. This holds especially for text and logos that are closely located to left and right borders of the screen.
A side effect of this stretching is that the aspect ratio of the content is somewhat distorted.

Table 13. Sizes of the "invisible" borders

Display	Pixels on the left	Pixels on the right
42" display, firmware version > 9.6	10	10
20" display, firmware version > 8.0	8	8

9 Converting depth to disparity

In 3D application a 3 dimensional (Cartesian) coordinate system is used to model objects. Normally these models are used to generate 2D images of scenes like: games or other applications. The function below describes the translation from depth (Z) to disparity (D(Z)). Herein depth describes the depth extracted from the application and needs to be normalized between 0 and 1. Further disparity refers to the difference in images from the left and right eye that the brain uses as a binocular cue to determine depth or distance of an object. The function below will describe the correct translation between depth and disparity:

$$D(Z) = M * \left(1 - \frac{vz}{Z - Zd + vz}\right) + C$$

where:

$D : disparity \quad [0,255]$

$Z : depth \quad [0,1]$

M : Linear function multiplier

Zd : depth of display plane

vz : View distance in coordinate units

C : Linear function constant

Within this formula there are a number of constants present M, Zd, vz and C. To obtain the best 3D performance for each of our displays use the correct values from the table below:

Constant	42”	20”
Zd	0.467481	0.459813
vz	7.655192	6.180772
M	-1960.37	-1586.34
C	127.5	127.5

Appendix A: CRC-32 implementation

crc32.h:

```
/*
------------------------------------------------------------------------------

  Philips 3D Solutions - Eindhoven

  CRC32 implementation adapted from public domain code written by Eric Durbin.

------------------------------------------------------------------------------
*/

#ifndef _CRC32_H
#define _CRC32_H

#ifdef __cplusplus
extern "C" {
#endif

/* crc32.h
   header file for crc32 checksum
*/

/* function prototypes */
unsigned long CalcCRC32(unsigned char *p, unsigned long len);

#ifdef __cplusplus
}
#endif

#endif /* _CRC32_H */
```

crc32.c

```
/*
------------------------------------------------------------------------------

  Philips 3D Solutions - Eindhoven

  CRC32 implementation adapted from public domain code written by Eric Durbin.
  Original header follows

-- START OF ORIGINAL HEADER ---------------------------------------------------

  crc32.c

   C implementation of CRC-32 checksums for NAACCR records.  Code is based
   upon and utilizes algorithm published by Ross Williams.

   This file contains:
      CRC lookup table
      function CalcCRC32 for calculating CRC-32 checksum
      function AssignCRC32 for assigning CRC-32 in NAACCR record
```

```
      function CheckCRC32 for checking CRC-32 in NAACCR record

   Provided by:
      Eric Durbin
      Kentucky Cancer Registry
      University of Kentucky
      October 14, 1998

   Status:
      Public Domain

-- END OF ORIGINAL HEADER -----------------------------------------------------
*/

#include "crc32.h"

/*****************************************************************/
/*                                                               */
/* CRC LOOKUP TABLE                                              */
/* ================                                              */
/* The following CRC lookup table was generated automagically   */
/* by the Rocksoft^tm Model CRC Algorithm Table Generation       */
/* Program V1.0 using the following model parameters:           */
/*                                                               */
/*    Width   : 4 bytes.                                         */
/*    Poly    : 0x04C11DB7L                                      */
/*    Reverse : FALSE.                                           */
/*                                                               */
/* For more information on the Rocksoft^tm Model CRC Algorithm,  */
/* see the document titled "A Painless Guide to CRC Error        */
/* Detection Algorithms" by Ross Williams                        */
/* (ross@guest.adelaide.edu.au.). This document is likely to be  */
/* in the FTP archive "ftp.adelaide.edu.au/pub/rocksoft".        */
/*                                                               */
/*****************************************************************/

unsigned long  crctable[256] =
{
 0x00000000L, 0x04C11DB7L, 0x09823B6EL, 0x0D4326D9L,
 0x130476DCL, 0x17C56B6BL, 0x1A864DB2L, 0x1E475005L,
 0x2608EDB8L, 0x22C9F00FL, 0x2F8AD6D6L, 0x2B4BCB61L,
 0x350C9B64L, 0x31CD86D3L, 0x3C8EA00AL, 0x384FBDBDL,
 0x4C11DB70L, 0x48D0C6C7L, 0x4593E01EL, 0x4152FDA9L,
 0x5F15ADACL, 0x5BD4B01BL, 0x569796C2L, 0x52568B75L,
 0x6A1936C8L, 0x6ED82B7FL, 0x639B0DA6L, 0x675A1011L,
 0x791D4014L, 0x7DDC5DA3L, 0x709F7B7AL, 0x745E66CDL,
 0x9823B6E0L, 0x9CE2AB57L, 0x91A18D8EL, 0x95609039L,
 0x8B27C03CL, 0x8FE6DD8BL, 0x82A5FB52L, 0x8664E6E5L,
 0xBE2B5B58L, 0xBAEA46EFL, 0xB7A96036L, 0xB3687D81L,
 0xAD2F2D84L, 0xA9EE3033L, 0xA4AD16EAL, 0xA06C0B5DL,
 0xD4326D90L, 0xD0F37027L, 0xDDB056FEL, 0xD9714B49L,
 0xC7361B4CL, 0xC3F706FBL, 0xCEB42022L, 0xCA753D95L,
 0xF23A8028L, 0xF6FB9D9FL, 0xFBB8BB46L, 0xFF79A6F1L,
 0xE13EF6F4L, 0xE5FFEB43L, 0xE8BCCD9AL, 0xEC7DD02DL,
 0x34867077L, 0x30476DC0L, 0x3D044B19L, 0x39C556AEL,
```

```
 0x278206ABL, 0x23431B1CL, 0x2E003DC5L, 0x2AC12072L,
 0x128E9DCFL, 0x164F8078L, 0x1B0CA6A1L, 0x1FCDBB16L,
 0x018AEB13L, 0x054BF6A4L, 0x0808D07DL, 0x0CC9CDCAL,
 0x7897AB07L, 0x7C56B6B0L, 0x71159069L, 0x75D48DDEL,
 0x6B93DDDBL, 0x6F52C06CL, 0x6211E6B5L, 0x66D0FB02L,
 0x5E9F46BFL, 0x5A5E5B08L, 0x571D7DD1L, 0x53DC6066L,
 0x4D9B3063L, 0x495A2DD4L, 0x44190B0DL, 0x40D816BAL,
 0xACA5C697L, 0xA864DB20L, 0xA527FDF9L, 0xA1E6E04EL,
 0xBFA1B04BL, 0xBB60ADFCL, 0xB6238B25L, 0xB2E29692L,
 0x8AAD2B2FL, 0x8E6C3698L, 0x832F1041L, 0x87EE0DF6L,
 0x99A95DF3L, 0x9D684044L, 0x902B669DL, 0x94EA7B2AL,
 0xE0B41DE7L, 0xE4750050L, 0xE9362689L, 0xEDF73B3EL,
 0xF3B06B3BL, 0xF771768CL, 0xFA325055L, 0xFEF34DE2L,
 0xC6BCF05FL, 0xC27DEDE8L, 0xCF3ECB31L, 0xCBFFD686L,
 0xD5B88683L, 0xD1799B34L, 0xDC3ABDEDL, 0xD8FBA05AL,
 0x690CE0EEL, 0x6DCDFD59L, 0x608EDB80L, 0x644FC637L,
 0x7A089632L, 0x7EC98B85L, 0x738AAD5CL, 0x774BB0EBL,
 0x4F040D56L, 0x4BC510E1L, 0x46863638L, 0x42472B8FL,
 0x5C007B8AL, 0x58C1663DL, 0x558240E4L, 0x51435D53L,
 0x251D3B9EL, 0x21DC2629L, 0x2C9F00F0L, 0x285E1D47L,
 0x36194D42L, 0x32D850F5L, 0x3F9B762CL, 0x3B5A6B9BL,
 0x0315D626L, 0x07D4CB91L, 0x0A97ED48L, 0x0E56F0FFL,
 0x1011A0FAL, 0x14D0BD4DL, 0x19939B94L, 0x1D528623L,
 0xF12F560EL, 0xF5EE4BB9L, 0xF8AD6D60L, 0xFC6C70D7L,
 0xE22B20D2L, 0xE6EA3D65L, 0xEBA91BBCL, 0xEF68060BL,
 0xD727BBB6L, 0xD3E6A601L, 0xDEA580D8L, 0xDA649D6FL,
 0xC423CD6AL, 0xC0E2D0DDL, 0xCDA1F604L, 0xC960EBB3L,
 0xBD3E8D7EL, 0xB9FF90C9L, 0xB4BCB610L, 0xB07DABA7L,
 0xAE3AFBA2L, 0xAAFBE615L, 0xA7B8C0CCL, 0xA379DD7BL,
 0x9B3660C6L, 0x9FF77D71L, 0x92B45BA8L, 0x9675461FL,
 0x8832161AL, 0x8CF30BADL, 0x81B02D74L, 0x857130C3L,
 0x5D8A9099L, 0x594B8D2EL, 0x5408ABF7L, 0x50C9B640L,
 0x4E8EE645L, 0x4A4FFBF2L, 0x470CDD2BL, 0x43CDC09CL,
 0x7B827D21L, 0x7F436096L, 0x7200464FL, 0x76C15BF8L,
 0x68860BFDL, 0x6C47164AL, 0x61043093L, 0x65C52D24L,
 0x119B4BE9L, 0x155A565EL, 0x18197087L, 0x1CD86D30L,
 0x029F3D35L, 0x065E2082L, 0x0B1D065BL, 0x0FDC1BECL,
 0x3793A651L, 0x3352BBE6L, 0x3E119D3FL, 0x3AD08088L,
 0x2497D08DL, 0x2056CD3AL, 0x2D15EBE3L, 0x29D4F654L,
 0xC5A92679L, 0xC1683BCEL, 0xCC2B1D17L, 0xC8EA00A0L,
 0xD6AD50A5L, 0xD26C4D12L, 0xDF2F6BCBL, 0xDBEE767CL,
 0xE3A1CBC1L, 0xE760D676L, 0xEA23F0AFL, 0xEEE2ED18L,
 0xF0A5BD1DL, 0xF464A0AAL, 0xF9278673L, 0xFDE69BC4L,
 0x89B8FD09L, 0x8D79E0BEL, 0x803AC667L, 0x84FBDBD0L,
 0x9ABC8BD5L, 0x9E7D9662L, 0x933EB0BBL, 0x97FFAD0CL,
 0xAFB010B1L, 0xAB710D06L, 0xA6322BDFL, 0xA2F33668L,
 0xBCB4666DL, 0xB8757BDAL, 0xB5365D03L, 0xB1F740B4L
};

/*********************************************************************/
/*                    End of CRC Lookup Table                        */
/*********************************************************************/

/* Calculate CRC-32 Checksum

   Uses reflected table driven method documented by Ross Williams.
```

```
   PARAMETERS:
     unsigned char *p           Buffer containing data over which
                                to calculate the CRC
     unsigned long len          Length of data in the buffer

   RETURNS:
     checksum value

   Author:
     Eric Durbin          1998-10-14

   Modified by:
     Philips 3D Solutions 2007-10-11

   Status:
     Public Domain
*/
unsigned long CalcCRC32(unsigned char *p, unsigned long len)
{
  unsigned long i;
  unsigned long crc = 0;

  /* process each byte prior to checksum field */
  for (i = 0; i < len; i++)
  {
    crc = crctable[(crc >> 24) ^ *p++] ^ (crc << 8);
  }

  return crc;
}
```