Description of the ULCD Library

Version 1.2

2011/07/14

The content of this document is highly confidential and should be handled accordingly.

CTR-06-0016-002-A
Released: October 3, 2011

Table of Contents

Figures

Equations

1 Introduction

The ULCD library supports three-dimensional viewing on CTR. This document describes the internal processes in the ULCD library and introduces sample implementations.

1.1 Objective

The ULCD library calculates the camera parameters that are required to render images that are seen from the left eye and the right eye. For input it uses camera parameters that were conventionally created and modified in an application (without taking stereoscopic representations into account). You can use the 3D depth slider on the CTR system to adjust the intensity of stereoscopic (3D) effects.

You can use this library to unify methods of representing stereoscopy across various applications.

1.2 Terminology Used in This Document

This document uses the following terminology.

Real Space

The distance from the player to the upper LCD and other factors are relevant to stereoscopic display on CTR. To differentiate between these and the space in the application, we use the term “real space” to indicate the space in which the player and the CTR system exist.

Virtual Space

Virtual space refers to the space created in the application, as opposed to real space.

Base Camera

The base camera is the one that the application sets and creates according to the scene. This information is required for the ULCD library to calculate cameras for the left eye and the right eye.

Base Plane

In virtual space, this plane ties images to the surface of the CTR LCD when stereoscopic display is enabled. It is one cross section of the camera viewing volume.

3D Depth Slider

This is the name of the 3D depth slider on the CTR system. In this library it is used to adjust the distance between the right and left cameras, which are generated in virtual space.

2 Overview of ULCD Library Processes

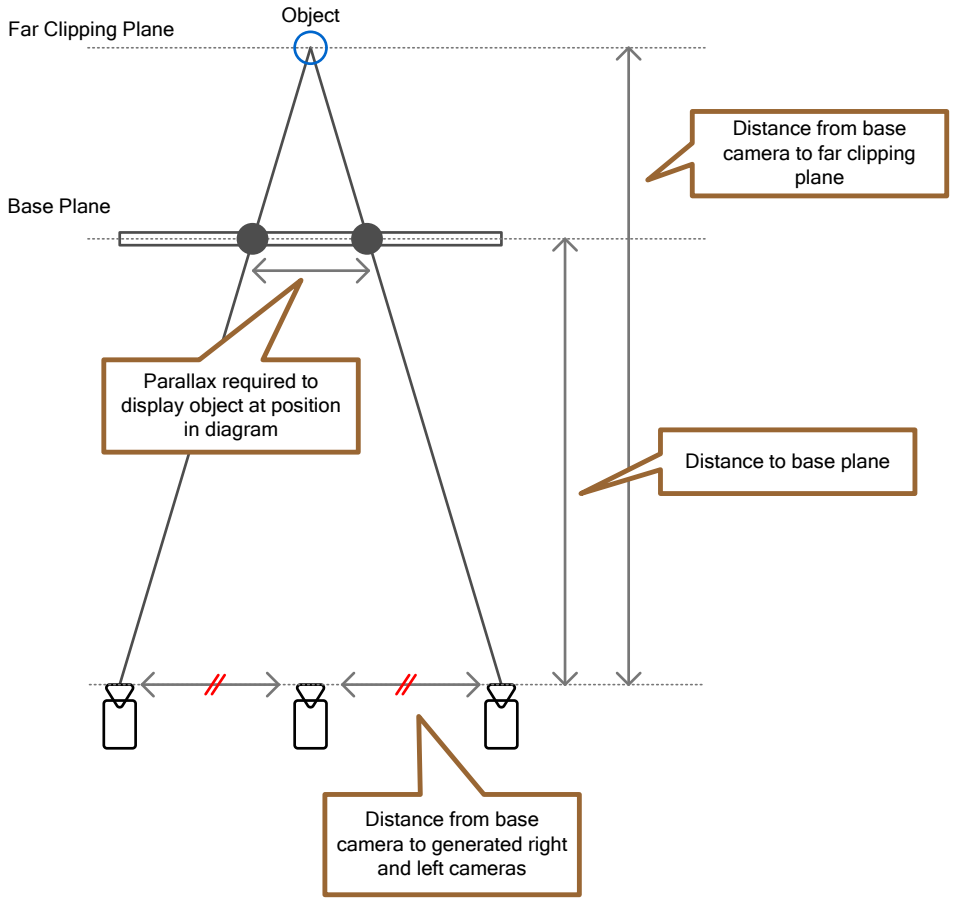
The ULCD library uses two methods to calculate the matrices used to generate the parallax images that are used for stereoscopic display. This section gives a simple description of these mechanisms. For further details, see Chapter 3 Principles.

2.1 Application Priority Method (Maintain Base Camera Settings Whenever Possible)

Figure 2-1 shows a general description of this procedure. With this method, preconditions (see Chapter 3 Principles) are used to obtain the parallax required to view an object on the far clipping plane. Left and right cameras are generated to fulfill these preconditions. The position of the specified base camera along the direction of the axis does not change, and thus neither does the viewing volume of the base camera.

This method adjusts the parallax to the maximum value that can be used to display an object positioned at the far clipping plane.

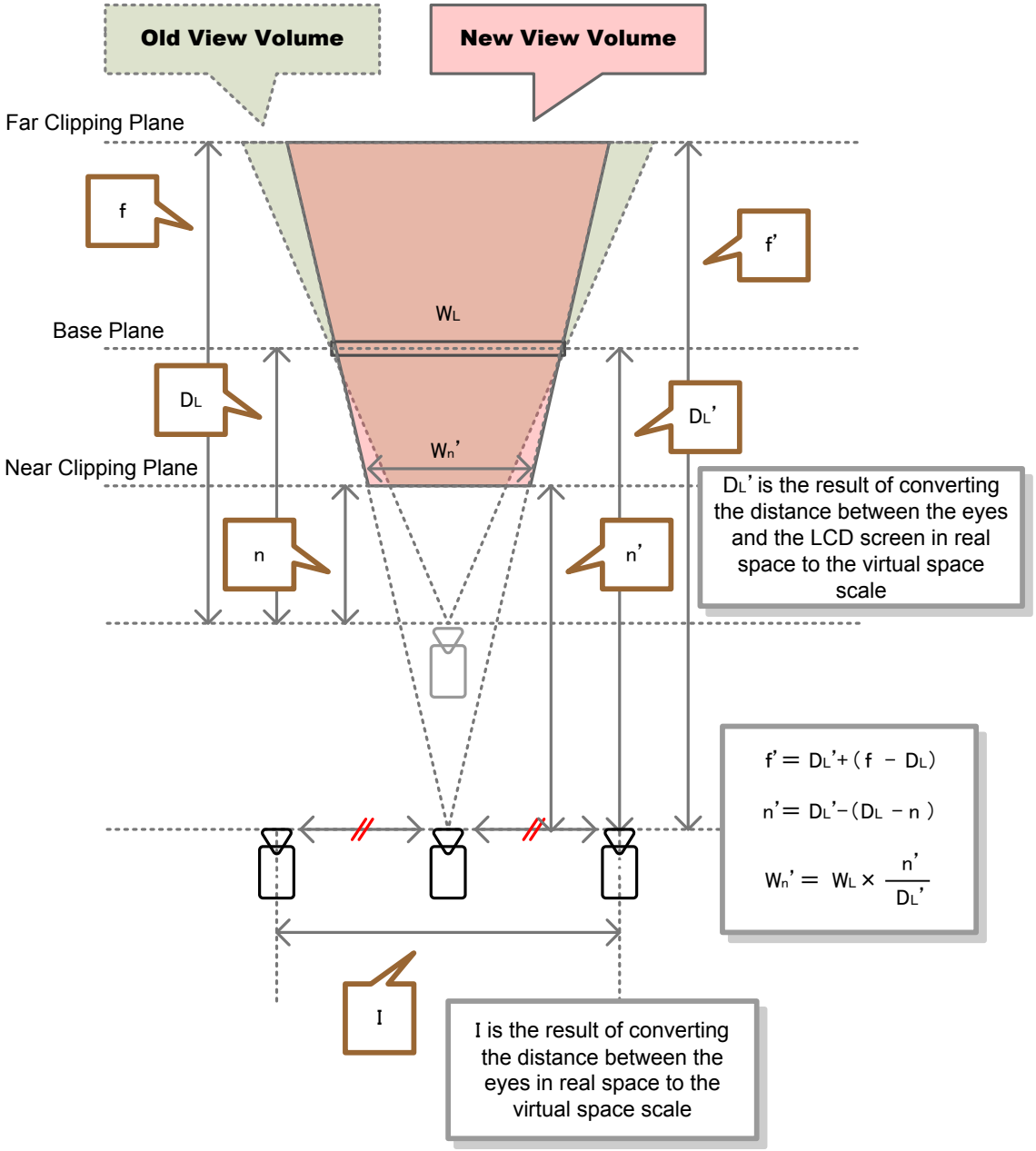
Figure 2-1 Application Priority Method (Overview)



2.2 Realism Priority Method (Automatically Change Base Camera Settings as Necessary)

Figure 2-2 shows a general description of this procedure. The symbols in the figure correspond to the descriptions in Chapter 3 Principles. With this method, the camera is adjusted in virtual space based on certain assumed conditions, such as the position of the CTR system with respect to the player. Therefore the position of the specified base camera along the direction of the axis is changed, and the viewing volume also changes as shown in the diagram.

Figure 2-2 Realism Priority Method (Overview)



3 Principles

3.1 Preconditions

Calculations have certain preconditions.

The following assumptions are made concerning elements related to the player and the CTR system in real space.

- The distance between the player's eyes is 62 mm $(Dist_{eye})$.
- The distance from the surface of the upper LCD on the CTR system to the player's eyes is 289 mm $(Dist_{e2d})$.

Based on information obtained in previous studies, we take $(Depth_{ltd})$ as the boundary distance out to which human eyes experience a natural-feeling sense of depth on the CTR system. The boundary parallax or *limit parallax* in real space required to represent this depth (in other words, the parallax required at the base plane) is referred to as P_{r_ltd}. **The maximum parallax values for forward and backward directions in depth from the player's point of view are set forth by the guidelines.**

The short side of the CTR system's upper LCD is 46.08 mm (Len_{disp}) long.

3.2 Calculating the Distance Between the Left and Right Cameras and Calculating Each Camera's Viewing Volume

The following input data is required.

1. A view matrix used as the base for generating parallax images for stereoscopy.
2. A projection matrix used as the base for generating camera viewing volumes for the left and right eyes.
3. The distance (D_{level}) in virtual space from the camera to a point which you wish to position on the surface of the LCD.
4. A coefficient (D_r) for adjusting the level of stereoscopy; this coefficient must be in the range $[0,1]$.

You can find the width and height of the base plane from D_{level} and the viewing volume parameters l_{base}, r_{base}, b_{base}, t_{base}, n_{base}, and f_{base} (left, right, bottom, top, near, far) that can be reverse calculated from the projection matrix specified as input data #2.

Equation 3-1 Base Plane Width

$$W_{level} = |r_{base} - l_{base}| \times \frac{D_{level}}{n_{base}}$$

Equation 3-2 Base Plane Height

$$H_{level} = |t_{base} - b_{base}| \times \frac{D_{level}}{n_{base}}$$

Obtain the coefficient for converting the real space scale to the virtual space scale from the base plane's width and height, and from the actual dimensions of the upper CTR LCD.

Equation 3-3 Conversion Coefficient from Real Space to Virtual Space

$$scale_{r2v} = \frac{H_{level}}{Len_{disp}}$$

This library supports two calculation methods depending on which stereoscopic viewing method you wish to prioritize. These are described in the following sections.

3.2.1 Application Priority Calculation Method

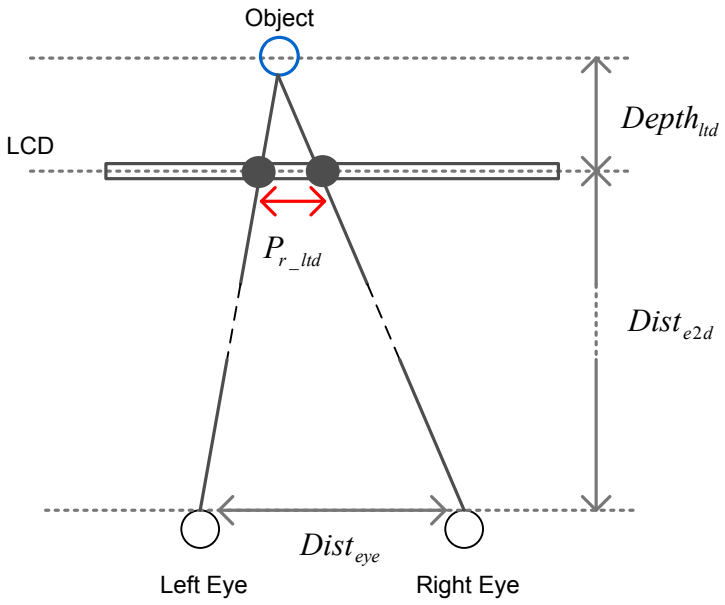
This method generates cameras to render images for the left eye and the right eye, maintaining as much as possible the view from the base camera (the view originally expected by the application).

The limit parallax in real space P_{r_ltd} (the greatest parallax that still feels natural) can be calculated from the preconditions in section 3.1 Preconditions. Figure 3-1 shows these relationships.

Equation 3-4 Limit Parallax in Real Space

$$P_{r\_ltd} = Dist_{eye} \times \left(\frac{Depth_{ltd}}{Dist_{e2d} + Depth_{ltd}} \right)$$

Figure 3-1 Limit Parallax in Real Space



This limit parallax is converted into the virtual space limit parallax P_{v_ltd}.

Equation 3-5 Limit Parallax in Virtual Space

$$P_{v\_ltd} = P_{r\_ltd} \times Scale_{r2v}$$

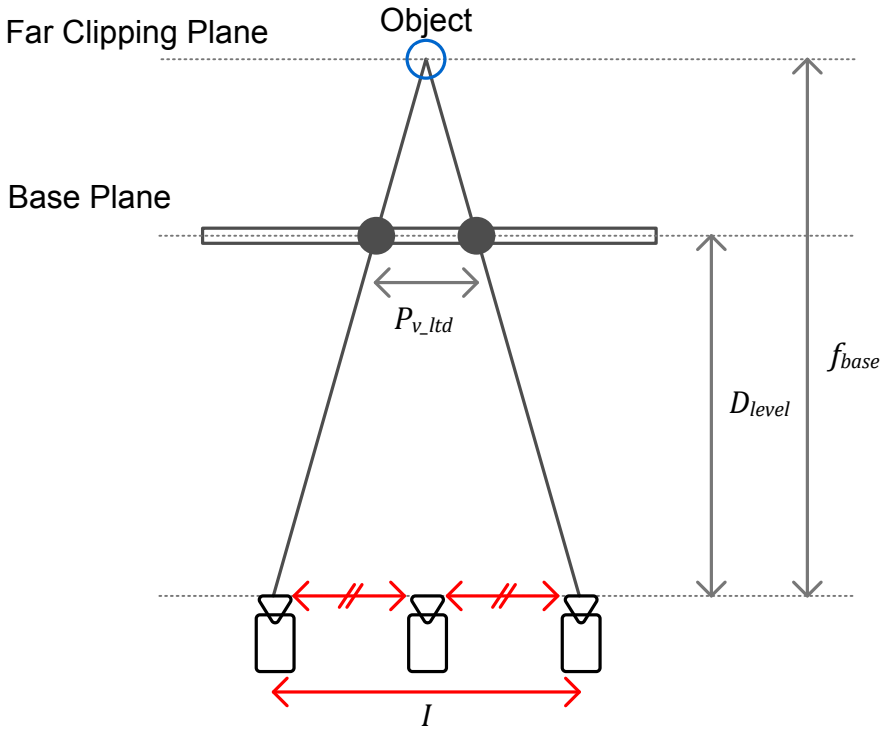
Without changing the depth position set for the base camera, the distance I between the cameras for the left and right eyes is found so that this limit parallax is the same as the parallax required on the base plane to display an object on the far clipping plane. Input data #3 is used to calculate this distance I. If the far clipping plane is in an abnormal position (for example closer than the base plane) this distance I will be 0.

The following equation is used. Figure 3-2 shows the relationships between each of the terms.

Equation 3-6 Distance Between the Left and Right Cameras

$$I = P_{v\_ltd} \times \left(\frac{f_{base}}{f_{base} - D_{level}}\right)$$

Figure 3-2 Distance Between the Left and Right Cameras in Virtual Space



In this method, parameters related to the viewing volume do not change because the position of the base camera is not changed.

Equation 3-7 Near Clipping Plane Width

$$W_n = |r_{base} - l_{base}|$$

Equation 3-8 Near Clipping Plane Height

$$H_n = |t_{base} - b_{base}|$$

3.2.2 Realism Priority Calculation Method

Based on the preconditions in section 3.1 Preconditions, in this method the relationship in real space between the player's eye spacing and the position of the CTR system's LCD is applied to the relationship in virtual space between the position of the two cameras (right and left) and the base plane. The parameters that have been set for the base camera are changed automatically as part of this process. By matching an object's appearance on the base plane with its appearance in real space, this method allows the player to view stereoscopic images naturally.

In other words, with this method the cameras are adjusted for the most natural view at a distance $Dist_{e2d}$ from the CTR LCD.

Use the equation below to find D_{level_new}. This is the "distance from the surface of the upper LCD on the CTR system to the player's eyes" given in section 3.1 Preconditions, converted to the scale of the virtual space.

Equation 3-9 Distance from the LCD to the Player's Eyes in Virtual Space

$$D_{level\_new} = Dist_{e2d} \times Scale_{r2v}$$

It is possible to obtain the distance to the newly generated camera's clipping planes n_{new} and f_{new} from the distance obtained above, input data #3, and the distance from the clipping planes obtained from input data #2.

Equation 3-10 Distance to the Newly Generated Camera's Clipping Planes

$$n_{new} = D_{level\_new} - (D_{level} - n_{base})$$
$$f_{new} = D_{level\_new} + (f_{base} - D_{level})$$

If Equation 3-10 results in either of the following situations, the clipping planes are adjusted as described in Equation 3-11 and Equation 3-12.

- The new near clipping plane is behind the camera (Equation 3-11)
- The far clipping plane is closer to the camera than the near clipping plane (Equation 3-12)

Equation 3-11 Adjusting a Near Clipping Plane That Is Behind the Camera

$$n_{new} = D_{level\_new} \times 0.01$$

Equation 3-12 Adjusting a Far Clipping Plane That Is Closer to the Camera Than the Near Clipping Plane

$$f_{new} = n_{new} \times 2.0$$

Because we are not moving the base plane, the base camera is moved forward or backward to satisfy the values we have obtained so far.

Next we find the dimensions of the near clipping plane (width, height), which have changed because it was moved. Also, we recalculate the near clipping plane range (left, right, top, bottom).

Equation 3-13 New Near Clipping Plane Width

$$W_{n\_new} = W_{level} \times \frac{n_{new}}{D_{level\_new}}$$

Equation 3-14 New Near Clipping Plane Height

$$H_{n\_new} = H_{level} \times \frac{n_{new}}{D_{level\_new}}$$

Equation 3-15 New Near Clipping Plane Range

$$l_{new} = tmp \times l_{base}$$
$$r_{new} = tmp \times r_{base}$$
$$t_{new} = tmp \times t_{base}$$
$$b_{new} = tmp \times b_{base}$$

In these equations, $= \frac{H_{n\_new}}{|t_{base} - b_{base}|}$.

Additionally, the distance between the player's eyes is applied to the distance I between the right and left cameras in virtual space.

Equation 3-16 Distance Between the Left and Right Cameras in Virtual Space

$$I = Dist_{eye} \times Scale_{r2v}$$

Figure 2-2 illustrates these equations.

3.3 Generating Projection Matrices

The adjustment coefficient D_r (input data #4) and the value of the CTR system's 3D depth slider are applied to the distance I between the left and right cameras, which was found in section 2.2 Realism Priority Method (Automatically Change Base Camera Settings as Necessary). There is no sense of depth at the lowest 3D depth slider value, for which the right and left cameras match the base camera.

Equation 3-17 Adjusted Distance Between the Left and Right Cameras

$$I = I \times vol \times D_r$$

In Equation 3-17, the 3D depth slider's input value is clamped to $vol = [0,1]$.

To get the respective viewing volumes for the left and right cameras, find the parallax P_n at the near clipping plane. Keep in mind that the left and right cameras are equidistant from the base camera position.

Equation 3-18 Parallax at the Near Clipping Plane

$$P_n = I \times 0.5 \times \left(\frac{D_{level} - n}{D_{level}}\right)$$

Note: The parameters of the base camera's viewing volume differ depending on whether the procedure described in section 3.2.1 Application Priority Calculation Method or in section 3.2.2 Realism Priority Calculation Method is used, but for convenience the following uniform notations are used: the near clipping plane width is W_n, the near clipping plane height is H_n, the distance from the base camera to the base plane is D_{level}, and the parameters related to the viewing volume are l, r, t, b, n, and f for the left, right, top, bottom, near, and far planes, respectively.

The positions of the near clipping plane in the left and right camera viewing volumes are found based on the parallax yielded by Equation 3-18. The top, bottom, near, and far planes are common between the left and right cameras because these cameras have only been moved horizontally.

Equation 3-19 Position of the Near Clipping Plane for the Left Camera

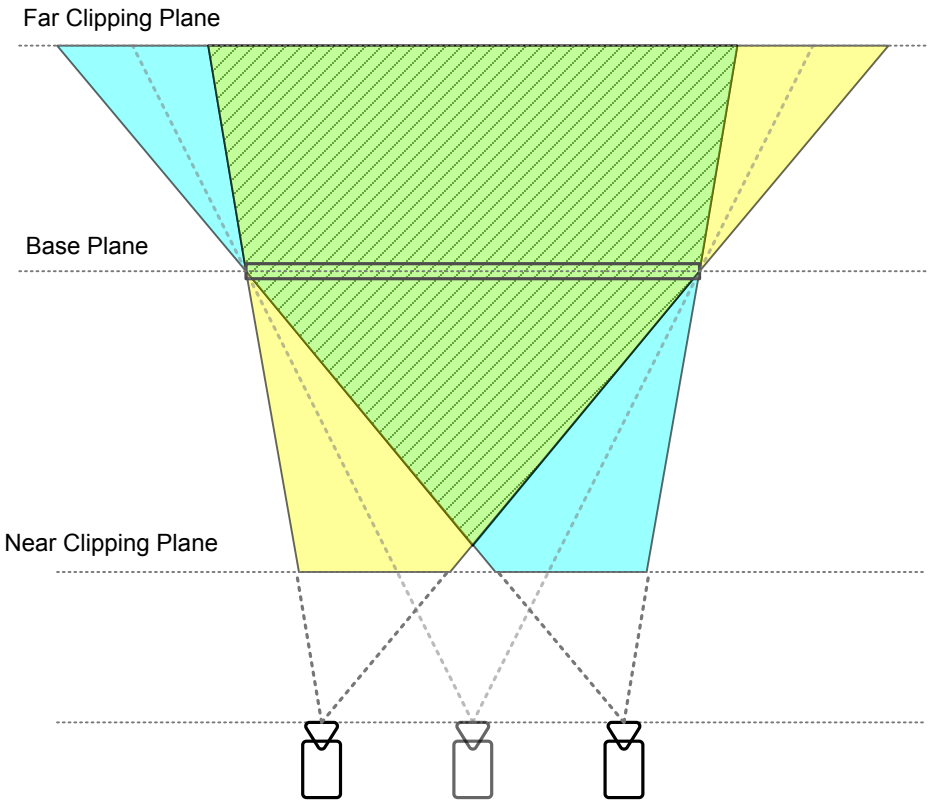
$$l_{left} = (l - P_n) + I \times 0.5$$
$$r_{left} = (r - P_n) + I \times 0.5$$

Equation 3-20 Position of the Near Clipping Plane for the Right Camera

$$l_{right} = (l + P_n) - I \times 0.5$$
$$r_{right} = (r + P_n) - I \times 0.5$$

The projection matrices for the left and right cameras are calculated based on parameters that follow from these equations. Figure 3-3 shows the viewing volumes yielded thus far. To generate a stereoscopic image, an object must be placed in the region where the viewing volumes intersect (shown by the portion of the figure shaded with diagonal lines).

Figure 3-3 Viewing Volumes for the Left and Right Cameras



3.4 Generating View Matrices

The left and right camera view matrices are each generated using the distance I between the left and right cameras, found in section 2.2 Realism Priority Method (Automatically Change Base Camera Settings as Necessary), and the base camera information derived from input data #1. (This base camera information consists of its position Pos_{base}, its direction Dir_{base}, and its right-hand direction E_{right}, which are all 3D vectors. Dir_{base} and E_{right} are unit vectors with a length of 1.)

However, when calculations prioritize realism, as described in section 2.2 Realism Priority Method (Automatically Change Base Camera Settings as Necessary), the left and right camera positions may have moved forward or backward (along the depth direction) relative to the original base camera. The base camera's position is therefore updated as follows.

Equation 3-21 Base Camera Position in Realism Priority Method

$$Pos_{base} = Pos_{base} - \left(D_{level\_new} - D_{level}\right) \times Dir_{base}$$

Because the base camera is midway between the left and right cameras, the positions of the left and right cameras relative to the base camera will be equal to the base camera's position plus or minus half of the distance between the left and right cameras. In the following equations, the left and right camera positions are Pos_{left} and Pos_{right}, and the direction of their look-at points are Tgt_{left} and

Tgt_{right}, respectively (we do not need to find the precise position of the look-at points because we only need to know the camera orientation).

Equation 3-22 Left Camera Position and Look-At Point

$$Pos_{left} = Pos_{base} - I \times 0.5 \times E_{right}$$
$$Tgt_{left} = Pos_{left} + Dir_{base}$$

Equation 3-23 Right Camera Position and Look-At Point

$$Pos_{right} = Pos_{base} + I \times 0.5 \times E_{right}$$
$$Tgt_{right} = Pos_{right} + Dir_{base}$$

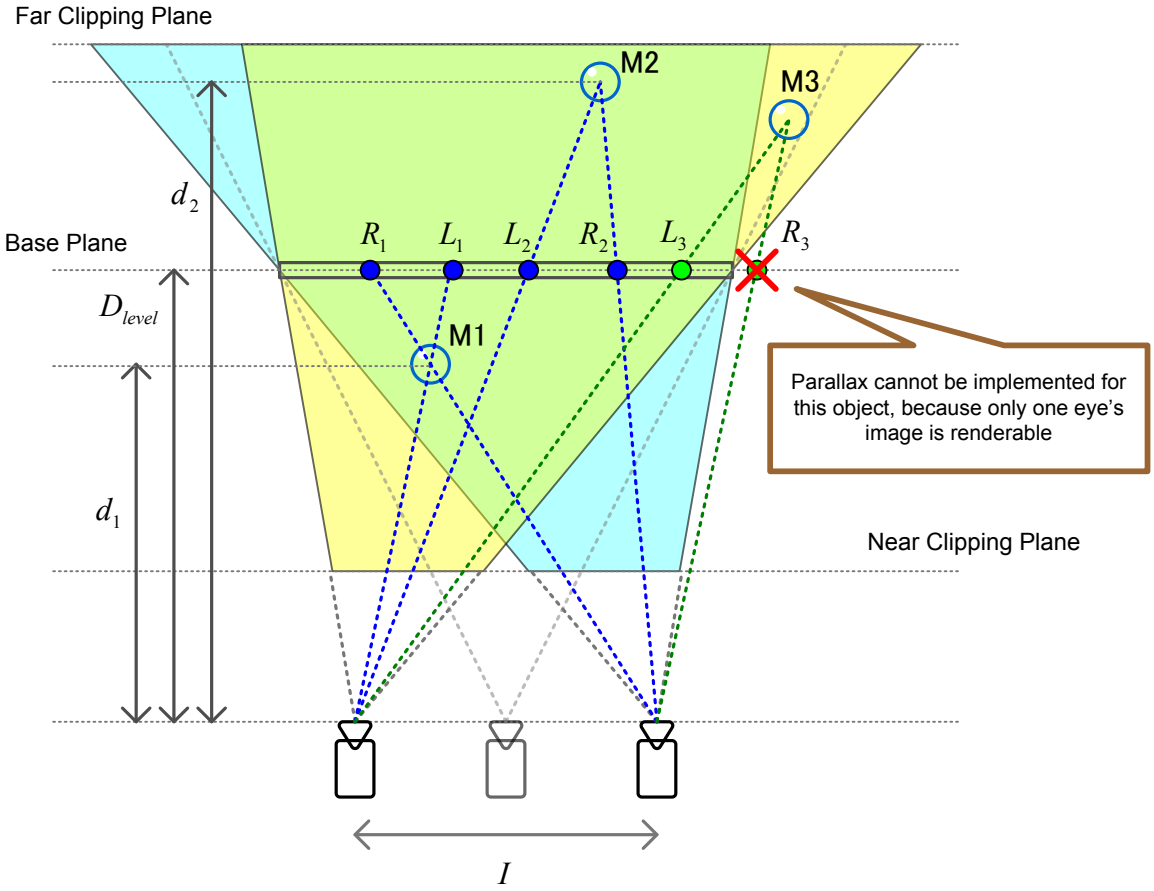
The left and right camera view matrices are generated from the above equations.

3.5 Parallax Required to Display an Object at an Arbitrary Position

Using the calculated matrices, you can implement stereoscopy without being aware of how much parallax is necessary for each 3D object. However, to display a 2D image, or a 3D object rendered using orthographic projection, you must create the left-eye and right-eye images yourself.

This library generates left and right cameras as shown in Figure 3-3 Viewing Volumes for the Left and Right Cameras. Figure 3-4 shows these cameras as well as the parallax required to display objects “in front of” the LCD screen (closer to the viewer than the screen) or “behind” the LCD screen (farther from the viewer than the screen).

Figure 3-4 Parallax Used to Display Objects at Arbitrary Positions



Object **M1** is positioned at distance d_1 from the cameras. To display **M1** closer to the viewer than the LCD is, render **M1** at the positions where straight lines connecting the cameras to **M1** would intersect the base plane. These positions are R_1 and L_1 for the right and left eyes, respectively. In other words, to display **M1** in front of the LCD, the parallax R_1L_1 is required. You can calculate the distance between R_1 and L_1 on the base plane by taking the distance I between the left and right cameras (found in section 3.2 Calculations for the Distance Between the Left and Right Cameras, and Each Camera's View Volume) and the distance D_{level} from the cameras to the base plane, and using trigonometry.

Equation 3-24 Parallax R_1L_1 for Objects in Front of the LCD Screen

$$R_1L_1 = I\frac{D_{level} - d_1}{d_1}, \qquad where\ D_{level} \geq d_1$$

Similarly, you can use the parallax R_2L_2 to display object **M2**, which is positioned at distance d_2 (behind the LCD).

Equation 3-25 Parallax R_2L_2 for Objects Behind the LCD Screen

$$R_2L_2 = I\frac{d_2 - D_{level}}{d_2}, \qquad where\ D_{level} < d_2$$

If an object is positioned like **M3** and therefore included in only one of the left/right camera viewing volumes, only one camera can pass a line through it and intersect the base plane. It is not possible to implement stereoscopic parallax for such objects.

3.6 Appropriate Parallax for Objects Located at the Maximum Possible Distance from the Base Plane

This section explains the appropriate degree to which the left- and right-eye images (whether of a 2D image or an object rendered using orthogonal projection) should be offset from each other to make that image or object appear to have the maximum possible stereoscopic depth, where it is located the maximum possible distance from the viewer.

Objects positioned at maximum possible depth can be represented by offsetting the left and right images by a distance equal to the limit parallax P_{r_ltd} (this distance is the actual offset on the surface of the LCD). In other words, to represent a far-off background with stereoscopic display enabled, the images rendered to the render buffers for the left and right eyes are each offset from the position of the base camera by half of the limit parallax.

In practice, however, you must use P_{v_ltd} (the result of converting P_{r_ltd} to the scale of virtual space) as the offset during rendering.

In principle, this technique can also be used to display stereoscopic images that appear to be jumping out in front of the LCD screen (toward the user) to the furthest extent possible. However, you must be careful when doing so, since objects that appear to be located in front of the screen do not have as wide a display range as those that appear to be located behind the screen. This difference in range is shown in Figure 3-4 Parallax Used to Display Objects at Arbitrary Positions.

4 Important Notes for Stereoscopic Images

Note: This document may be revised or expanded to comply with guidelines that are being reviewed separately.

4.1 Placing Objects in Front of the LCD Screen

When you use stereoscopic display to position objects so that they appear to jump out in front of the surface of the LCD, you must position them so that they do not touch any edge of the LCD screen. If an object does overlap the edge of the screen, then the screen's border, which appears (from the viewer's point of view) to be behind the object, will cover up an object that is supposed to be in front of it. This causes an unnatural effect, making it impossible for the player to experience the correct 3D sense of depth.

You must also be careful when positioning the near clipping plane. Normally the near clipping plane is very close to the camera. However, this usual distance to the near clipping plane does not match the boundary distance from the screen toward the player that allows the player to experience the correct 3D sense of depth. Therefore, when an object gets too close to the camera, stereoscopic display of the object becomes impossible.

This is not a problem that can be solved by simply placing the near clipping plane far from the camera. In each individual application you must consider and apply ways to prevent objects from getting too close to the camera.

4.2 Positioning When 2D Objects and 3D Objects Are Displayed Together

It is possible to display 2D and 3D objects at the same time by combining orthographic projection with objects rendered at camera settings that vary in a way that mimics perspective projection. When you do this, you must adjust the position, size, and order with which you render the 2D objects to ensure that the 2D and 3D objects have the proper foreground/background relationship.

If you use perspective projection to render the 2D objects as well as the 3D objects, you do not have to adjust the 2D objects themselves. However, you must still consider the foreground/background relationship of the 3D and 2D objects when deciding where to position your 2D objects, because 2D objects could be unintentionally hidden by 3D objects.

NW4C's LYT library provides an `orthoStereo` demo as a sample implementation of using the `StereoCamera::GetCoefficientForParallax` function to display stereoscopic images with orthographic projection.

5 StereoCamera Class

The ULCD library provides the `nn::ulcd::StereoCamera` class for the information described in Chapter 3 Principles. This following chapter describes the member functions of the `StereoCamera` class (the "`nn::ulcd::StereoCamera`" portion is omitted).

Note: The specifications for this class may be revised to comply with stipulations in the guidelines.

5.1 Initialization and Shutdown

```
void Initialize(void);
void Finalize(void);
```

Initializes and finalizes the `StereoCamera` class.

The `Initialize` function initializes all internal member variables to 0. You must call the `Finalize` function to destroy an instance of the `StereoCamera` class.

5.2 Setting the Base Camera

```
void SetBaseFrustum(const nn::math::Matrix44 *proj);
void SetBaseFrustum(f32 left, f32 right, f32 bottom, f32 top, f32 near, f32 far);
```

Specifies the parameters that comprise the base camera's viewing volume.

Either specify the viewing volume parameters directly or input the projection matrix created by the `nn::math::MTX44Frustum` or `nn::math::MTX44Perspective` function. These parameters correspond to input data #2 in section 3.2 Calculating the Distance Between the Left and Right Cameras and Calculating Each Camera's Viewing Volume. If you specify a projection matrix in `nn::math::Matrix44` format, the base frustum will not be calculated correctly unless the projection matrix was calculated based on the definition of the viewing volume of the CTR graphics system (specifically, the z-coordinate must be clipped to the range $[0, -w_c]$).

```
void SetBaseCamera(const nn::math::Matrix34 *view);
void SetBaseCamera(nn::math::Vector3 *position, nn::math::Vector3 *rightDir,
                   nn::math::Vector3 *upDir, nn::math::Vector3 *targetDir);
```

Specifies the base camera's view matrix.

Either specify the view matrix created by a function such as `nn::math::MTX34LookAt` or directly specify the parameters for the base camera's position. This corresponds to input data #1 in section 3.2. Make sure to use the right-handed coordinate system adopted by the CTR graphics system when specifying the view matrix or the `nn::math::Vector3` arguments.

5.3 Calculating the Left and Right Cameras

The following function is used if you select the method described in section 3.2.1 Application Priority Calculation Method.

```
void CalculateMatrices(nn::math::Matrix44* projL, nn::math::Matrix34* viewL,
                       nn::math::Matrix44* projR, nn::math::Matrix34* viewR,
                       const f32 depthLevel, const f32 factor,
                       const nn::math::PivotDirection pivot =
                                              nn::math::PIVOT_UPSIDE_TO_TOP);
```

The following function is used if you select the method described in section 3.2.2 Realism Priority Calculation Method.

```
void CalculateMatricesReal(nn::math::Matrix44* projL, nn::math::Matrix34* viewL,
                           nn::math::Matrix44* projR, nn::math::Matrix34* viewR,
                           const f32 depthLevel, const f32 factor,
                           const nn::math::PivotDirection pivot =
                                                 nn::math::PIVOT_UPSIDE_TO_TOP);
```

Both functions take the same arguments.

These functions calculate projection and view matrices that are stored in *projL*, *projR*, *viewL*, and *viewR*, respectively. The matrix calculations use the `nn::math` library.

Set *depthLevel* to the distance from the base camera to the plane containing points that you want to position on the surface of the LCD (the base plane). This corresponds to input data #3 in section 3.2 Calculating the Distance Between the Left and Right Cameras and Calculating Each Camera's Viewing Volume.

The *factor* argument is used to correct internally calculated results. If this argument is set equal to 0, there is zero parallax; if this argument is set equal to 1, there is zero correction of the results. This argument corresponds to input data #4 in section 3.2 Calculating the Distance Between the Left and Right Cameras and Calculating Each Camera's Viewing Volume.

These functions generate both a projection matrix and a rotation matrix. The projection matrix is multiplied by the rotation matrix. The *pivot* argument affects the rotation matrix. Following multiplication by the rotation matrix, the camera's upward direction will match the direction of *pivot*. If left unspecified, this argument is `nn::math::PIVOT_UPSIDE_TO_TOP`. Set this argument equal to `nn::math::PIVOT_NONE` if rotation is unnecessary.

5.4 Utilities

5.4.1 Utilities Valid Before CalculateMatrices * Is Called

5.4.1.1 SetLimitParallax

```
void SetLimitParallax(f32 limit);
```

Sets the limit parallax (in mm). The limit parallax can be set to any value between 0 and the maximum parallax value along the depth direction as set forth by the guidelines (see **Note** below). This value is applied to the results of the `CalculateMatrices` function. Specifically, if the application priority method is used (see section 3.2.1 for more details), objects located at the far clipping plane are offset along the surface of the LCD screen by the parallax value set using this function.

Note: As of CTR-SDK 0.13, specifying a value higher than the limit parallax causes the function to output a warning, but the function will still use this value.

5.4.1.2 GetLimitParallax

```
f32 GetLimitParallax(void) const;
```

Gets the limit parallax that was set using the `SetLimitParallax` function. If `SetLimitParallax` has not yet been called, by default this function will return the maximum limit parallax along the depth direction that the guidelines allow.

5.4.2 Utilities Valid After CalculateMatrices* Is Called

All of the following functions process the results of the most recent call to the `CalculateMatrices` or `CalculateMatricesReal` function (except when the `Initialize` function has initialized the internal state of the library).

5.4.2.1 GetParallax

```
f32 GetParallax(f32 distance) const;
```

Calculates the parallax required to place an image at a position separated from the camera by *`distance`* (the parallax measured from the display position as seen by the base camera, in other words, half of the parallax for the images for the left and right eyes), and then returns the ratio of this parallax to the LCD screen width.

You can multiply this function's return value by the LCD screen resolution (its width in pixels) to get the number of pixels by which to offset the left and right images that you create.

The return value is positive if *`distance`* is behind the base plane and negative if *`distance`* is in front of the base plane. The return value is 0 if *`distance`* is less than or equal to 0.

Note: As shown by object **M3** in Figure 3-4 of section 3.5 Parallax Required to Display an Object at an Arbitrary Position, proper stereoscopic representations are not possible if an object does not fall within both the left and right cameras' viewing volumes. Take care when using this function, because this function does not detect such situations.

5.4.2.2 GetMaxParallax

```
f32 GetMaxParallax(void) const;
```

Returns the ratio of the limit parallax to the width of the screen. This limit parallax is the parallax used to make an object appear with the greatest possible depth, as explained in section 3.6 Appropriate Parallax for Objects Located at the Maximum Possible Distance from the Base Plane. This ratio is based on the object's position as seen from the base camera, as is the case with the `GetParallax` function. Multiply this ratio by the LCD screen resolution (its width in pixels) to get the number of pixels by which to offset the left and right images.

5.4.2.3 GetDistanceToLevel

```
f32 GetDistanceToLevel(void) const;
```

Gets the distance from the cameras to the base plane. This is mainly used after the `CalculateMatricesReal` function automatically moves the cameras along the depth axis.

5.4.2.4 GetDistanceToNearClip

```
f32 GetDistanceToNearClip(void) const;
```

Gets the distance from the cameras to the near clipping plane. This is mainly used after the `CalculateMatricesReal` function automatically moves the cameras along the depth axis.

5.4.2.5 GetDistanceToFarClip

```
f32 GetDistanceToFarClip(void) const;
```

Gets the distance from the cameras to the far clipping plane. This is mainly used after the `CalculateMatricesReal` function automatically moves the cameras along the depth axis.

5.4.2.6 GetCoefficientForParallax

```
f32 GetCoefficientForParallax(void) const;
```

You don't normally need to use this function.

To display stereoscopic images with orthographic projection, you must calculate the parallax for each object separately. This function is provided to be run in the vertex shader to optimize some of the calculations used by the `GetParallax` function.

Multiplying this function's return value with $(D_{level} - d_1)/d_1$ from the following equation (in section 3.5 Parallax Required to Display an Object at an Arbitrary Position) results in the same value that is returned by the `GetParallax` function.

$$R_1L_1 = I\frac{D_{level} - d_1}{d_1}$$

See the `orthoStereo` demo in NW4C's LYT library for more information on how to apply this function.

Revision History

Version	Revision Date	Description
1.2	2011/07/14	• Corrected a typo in the function name in section 5.4.2.6 GetCoefficientForParallax.
1.1	2011/06/20	• 4.2 Positioning When 2D Objects and 3D Objects Are Displayed Together Added information on displaying stereoscopic images with orthographic projection. • 5.4.2 Utilities Valid After CalculateMatrices* Is Called Added the `GetCoefficientForParallax` function.
1.0	2010/09/17	• Revised text in section 3.1 Preconditions now that parameters have been decided. • In Chapter 5 StereoCamera Class, revised types for the arguments for each function, and revised descriptions of the `SetLimitParallax` and `GetLimitParallax` functions in line with changes to the maximum parallax values for forward and backward directions in depth.
0.5	2010/08/18	• Changed the description of the `Finalize` function in section 5.1 Initialization and Shutdown.
0.4	2010/06/16	• Changed the term "3D slider" to "3D depth slider" in sections 1.1 Objective and 1.2 Terminology Used in This Document. • Clarified that the maximum boundary parallax is set forth in the guidelines in section 3.1 Preconditions. • Added a restriction on the coefficient used to adjust the stereoscopic effect, and modified the subtraction operations in the formulas to use the absolute value in order to remove the dependency between the left/right and top/bottom values that specify the viewing volume in section 3.2 Calculating the Distance Between the Left and Right Cameras and Each Camera's View Volume. • Revised the equation for calculating the new viewing volume in section 3.3 Generating Projection Matrices. • Added a formula for finding the parallax required to display an object at an arbitrary position in section 3.5 Parallax Required to Display an Object at an Arbitrary Position. • Overhauled section 3.6 Appropriate Parallax for Objects Located at the Maximum Possible Distance from the Base Plane to reflect specification changes. • Added restrictions on the matrices that can be specified in section 5.2 Base Camera Settings. • Revised section 5.4 Utilities based on the addition of the `SetLimitParallax` and `GetLimitParallax` functions. • Added section 5.4.1 Utilities Enabled before CalculateMatrices * is Called, and added a section header for 5.4.2 Utilities Enabled after CalculateMatrices* is Called.
0.3	2010/05/07	• Changed some of the numbers in section 3.1 Preconditions.
0.2	2010/04/14	• Added Figure 3 2 Distance Between the Left and Right Cameras in Virtual Space and Figure 3 3 View Volume for the Left and Right Cameras. • Added section 3.5 Parallax Required to Display Objects at Any Position. • Added Chapter 5 StereoCamera Class. • Revisited overall terminology and other issues.
0.1	2010/03/11	• Initial version.