Warping

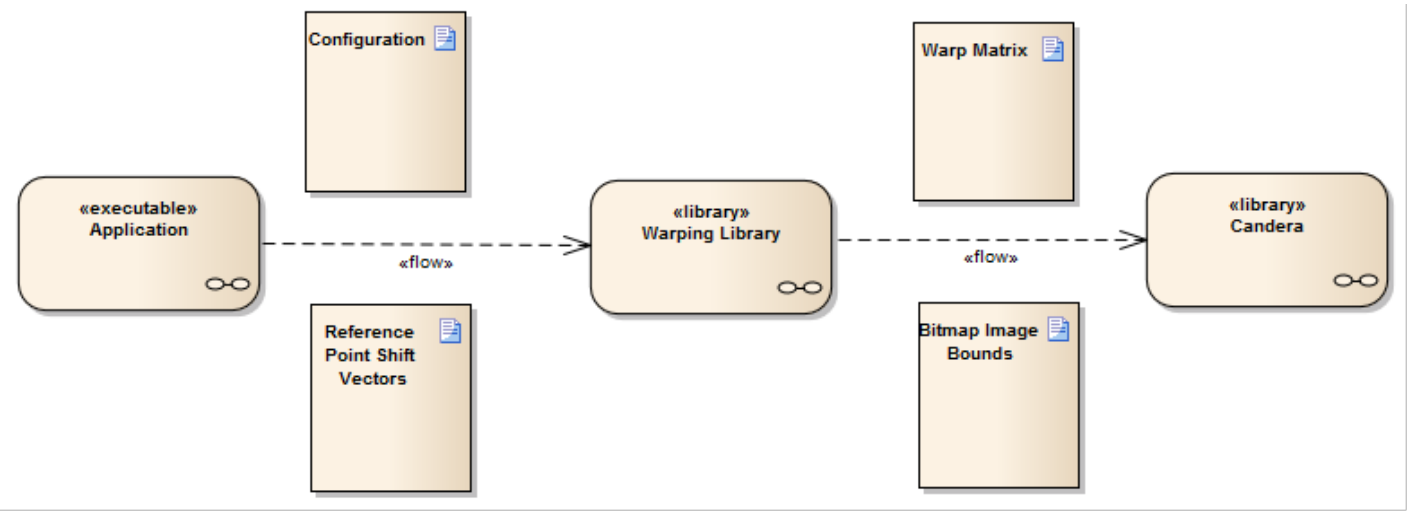
- Introduction
- Configuration
- Default Adjustments
- Enabling Warping
- Warping Sequence
- Warping Library API
- Warping Candera API

Introduction

Image warping is the process of digitally manipulating an image such that shapes portrayed in the image have been distorted. Warping may be used for correcting image distortion as well as for creative purposes (e.g., morphing). The same techniques are equally applicable to video. CGI Studio warping is intended to be used for correction image distortion only. The following two components are involved in the process:

- Warping Library: compute the warping matrix
- Candera: maps a displays image as texture to a mesh created from the warping matrix information and displays the correspondingly distorted result.

Following is a diagram depicting the information flow for warping:



Configuration

Description

Configuration information passed to the warping library is used to check constraints and as basis for computations.
Following is a description of the warping configuration parameters:

Reference Points

The reference points are represented by a 2D array with coordinates inside the display area to be warped. There are **NumOfRefPointsX** in horizontal direction and **NumOfRefPointsY** in vertical direction, both having values in the range 0..255. These are equally distributed over Bitmap-pixel.
Example:

- NumOfRefPointsX: 21 (0x15)
- NumOfRefPointsY: 11 (0x0B)

The number of reference points used influence to output of the distorted image with regard to Bitmap size. The output image will be less jagged the higher the number of reference points are used for warping computation

PixResolution

The Resolution of warping parameters determinine the warping accuracy. The final displacement of the reference points is the product of warping parameter by parameter resolution. Therefore, a warping parameter is given by the desired displacement divided by resolution.

The warping library supports the following pixel:

- 0: $^{1}/_{16}$ px
- 1: $^{1}/_{8}$ px
- 2: ¼ px
- 3: ½ px
- 4: 1 px
- 5: 2 px
- 6: 4 px
- 7: 8 px

Example:
If a texture point at (0,0) should be mapped at (10,10) the corresponding X[0] and Y[0] coordinates can be calculated by dividing the desired displacement by pixel resolution.
For a pixel resolution of $^{1}/_{16}$ px (0.0625):

```
X[0] = Xnew[0] – Xold[0] / 0.0625 = 160 (0xA0)
Y[0] = Ynew[0] –Yold[0] / 0.0625 = 160 (0xA0)
```

For a pixel resolution of ¼ px (0.25):

```
X[0] = Xnew[0] – Xold[0] / 0.25 = 40 (0x28)
Y[0] = Ynew[0] –Yold[0] / 0.25 = 40 (0x28)
```

NumOfBitsPerParam

Represents the number of bits used for each parameter (n-bit representation = bit length of each parameter).
The value should be in the range is 1..16.
Example:

- If NumOfBitsPerParam is 8 the warping parameters corresponding to X[0]Y[0] are represented as 0xA0 0xA0.
- If NumOfBitsPerParam is 12 the warping parameters corresponding to X[0]Y[0] are represented as 0x0A 0x00 0xA0.

> The **NumOfBitsPerParam** and **PixResolution** are represented on 8 bits, upper three bits are used for NumOfBitsPerParam and lower bit for PixResolution.

Example::

- 0x96 -> NumOfBitsPerParam: 12, PixResolution 0 ($^{1}/_{16}$ px)
- 0x98 -> NumOfBitsPerParam: 12, PixResolution 2 (¼ px)

DisplaySizeX and DisplaySizeY

Width and height data (in pixel) of the display hardware (range: 0..65535).

Example:

- 0x01 0xE0 – DisplaySizeX – 480px width
- 0X00 0XF0- DisplaySizeY – 240 px height

BitmapAreaX and BitmapAreaY

Width and height data (in pixels) of source surface used as input for warp. The reference points are equally distributed over this range in x and y direction with values in the range 0..65535.

Example:

- 0x01 0x98 – BitmapAreaX – 408px width
- 0X00 0XBE- BitmapAreaY– 190 px height

> Condition:
> BitmapAreaX≤ DisplaySizeX
> BitmapAreaY≤DisplaySizeY

BitmapAreaCenter

The Bitmap Area Center Point describes the center point of the image area used by ABK.

BitmapAreaShift

The Bitmap Area Shift describes the connection vector between the origin of the bitmap coordinate system (0,0) and the center point of the bitmap area - i.e. the Bitmap Area Center Point.

BitmapAreaShiftX and BitmapAreaShiftY

Shift values of bitmap area center point in horizontal and vertical direction (range 0..65535).

- ”0” BitmapAreaShiftX means the application window starts at the left edge on the display.
- ”0” BitmapAreaShiftY means the application window starts at the upper edge on the display.

Example:

- 0x00 0xCC – BitmapShiftAreaX – 204px
- 0X00 0X5F- BitmapShiftAreaY– 95 px

> Condition:
> (BitmapAreaY/2+BitmapAreaShiftY≤DisplaySizeY) AND (BitmapAreaY/2–BitmapAreaShiftY ≥0
> (BitmapAreaX/2+BitmapAreaShiftX≤DisplaySizeX) AND (BitmapAreaX/2–BitmapAreaShiftX ≥0

IlluminatedDisplayAreaX and IlluminatedDisplayAreaY

Width and height data (in pixel) for illuminated area. This represents the region in which the warped surface will be rendered. If the source surface size extends beyond the IlluminatedDisplayArea it will be scaled down with regards to display size.

Example:

- 0x01 0xE0 – IlluminatedDisplayAreaX – 480px width
- 0X00 0XF0- IlluminatedDisplayAreaY– 240 px height

> Condition:
> IlluminatedDisplayAreaX ≤ DisplaySizeY
> IlluminatedDisplayAreaY ≤ DisplaySizeY

IlluminatedDisplayAreaShiftX and IlluminatedDisplayAreaShiftY

Shift values of illuminatedDisplay area center point in horizontal and vertical direction (range 0..65535).

Example:

- 0x00 0xF0 – BitmapShiftAreaX – 240px
- 0X00 0X78- BitmapShiftAreaY– 120 px

> Condition:
> (IllDisplayAreaX/2+IllDisplayAreaShiftX≤DisplaySizeX) AND (IllDisplayAreaX/2–IllDisplayAreaShiftX ≥0)
> (IllDisplayAreaY/2+IllDisplayAreaShiftY≤DisplaySizeY) AND (IllDisplayAreaY/2–IllDisplayAreaShiftY ≥0)

RotationValue

The rotation value is represented in grad *100 with values in the range -18000 .. +18000. This is the default rotation value which will be applied to the warp image.

The rotation takes place around the warped center point of the bitmap area used. With an uneven number of reference points in x- and y-direction, this point coincides with the middle reference grid point, and the coordinates of the rotation axis are present. If an even number of reference grid points is used, the coordinates of the warped center point of the image are be calculated.

Example:

- 0x01 0xF4 – RotationValue: 500/100 = 50

ParamSet

Set of reference point shift vectors column-ordered.

The warping parameter correspond to the shift vectors (in pixels) of warping interpolation points in warped images in comparison to reference points in non-warped images (divided by the specified resolution). To save memory space and computing time, the absolute coordinates are not saved as warping parameters, but rather only the distance from the reference points (without rotation and shifting).

ParamSetLength

Length of ParamSet in bytes. This is calculated by:

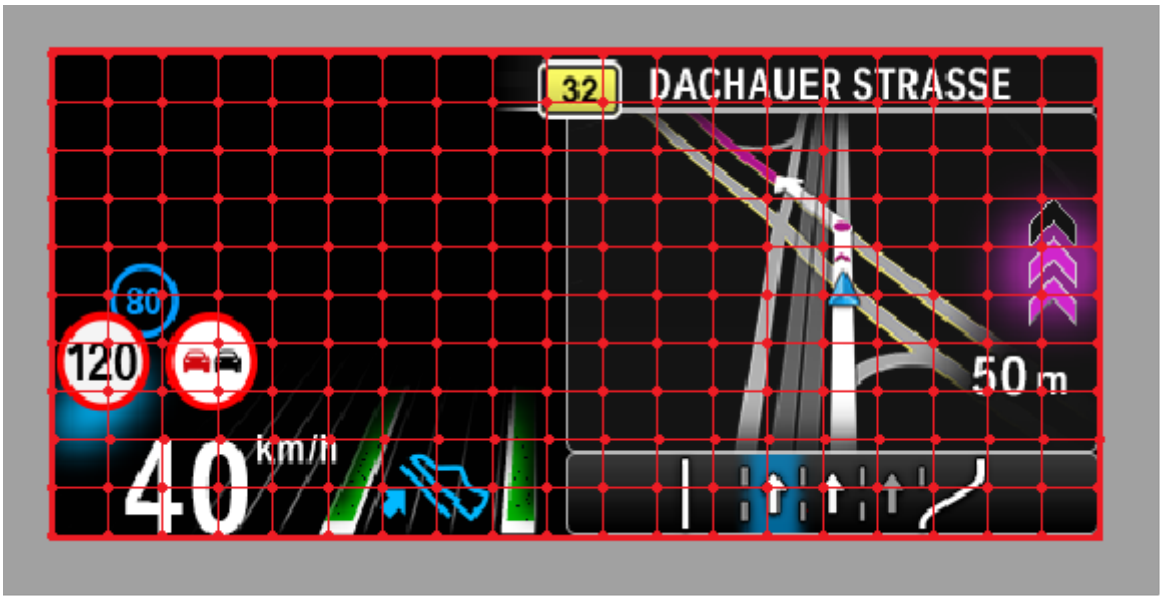
```
ParamSetLength = NumRefPointsX * NumRefPointsY * 2 * NumBitsPerParam
```

RotationLimit

This value represents the maximal rotation angle in grad – both positive and negative values are accepted. If the RotationLimit value is chosen such that at maxim rotation point, the image will extend beyond display size, it will be scaled down to the size that fits inside the display area when a rotation equal to RotationLimit is applied.

The *display area* comprises the whole image including the grey border.
The *bitmap image* area is the area within the grey border. This is the area we are interested in and which should be warped.
The reference points are evenly distributed over the bitmap image area.

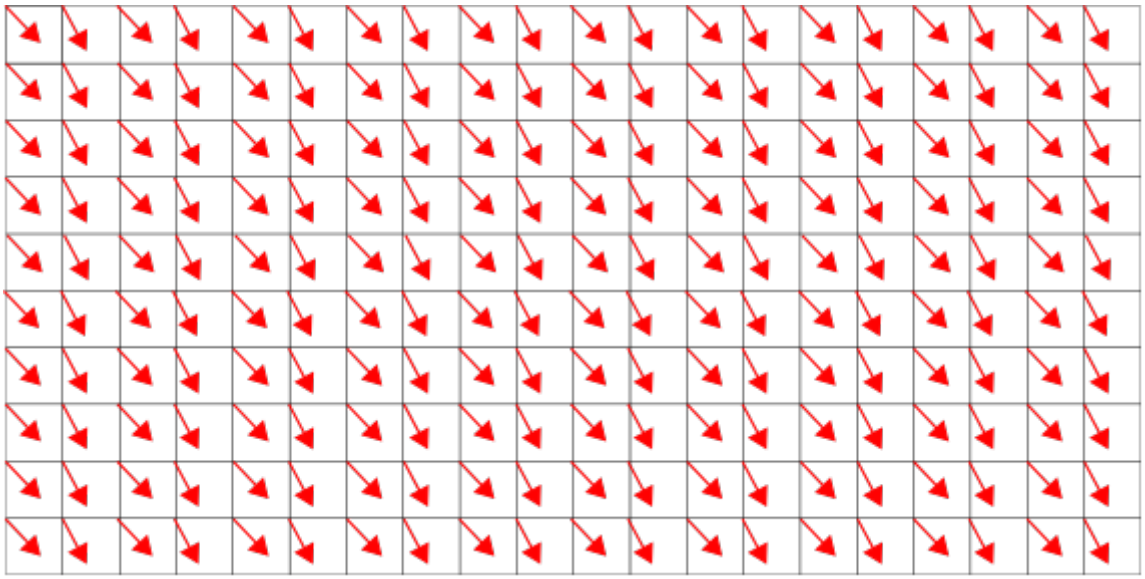
> The illuminated display area is the part of the display, which is backlit and projected to the windshield.



Reference Point Shift Vectors

The warping parameters are based on reference point. These points are mapped by a 2D array containing coordinates inside the display area to be warped. For ++-each reference point, a shift vector is stored by the application. Every shift vector determines the amount of distortion to be applied to the corresponding point in x- and y-direction. Figure 2 shows a reference point grid with corresponding shift vectors.

To preserve memory, the shift vectors are stored in a compressed format. The format is determined by configuration information passed to warping library (number of bits used, value resolution) and is used by warping library to de- and encode shift vectors. The decoded shift vectors are cached in the warping library.



Warp Matrix

The warp matrix is computed by the warping library. It is a 2-dimensional matrix. Each point corresponds to a reference point. The value of each point is the vector-addition of the reference point and corresponding shift vector. All values of the warp matrix are normalized to the range [0..1].

Warp Image Bounds

Warp image bounds can be computed by the warping library. Warp image bounds comprise upper left corner point of the bitmap image area, its width, and height (normalized to the range [0..1]).

They are used to specify to Candera which part of the rendered image shall be mapped to the warping mesh.

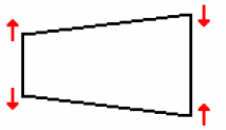
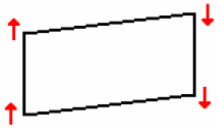
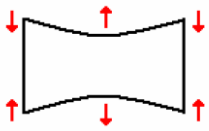
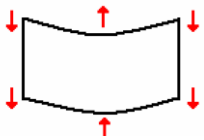
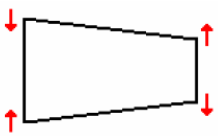
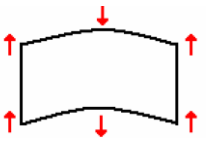
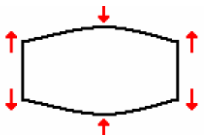
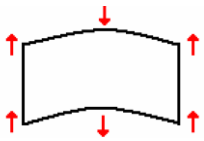
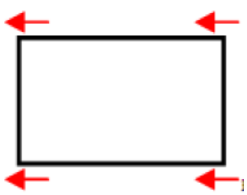
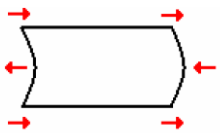
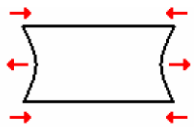
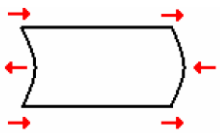
Default Adjustments

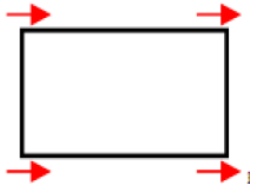
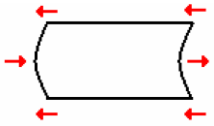
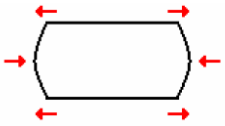
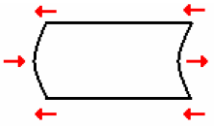
An adjustment is specified by a mode and a delta. Mode defines the way of the distortion, delta defines the amount. The following modes are available:

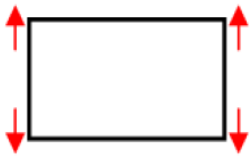
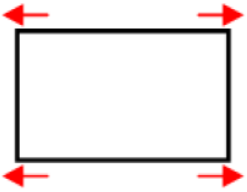
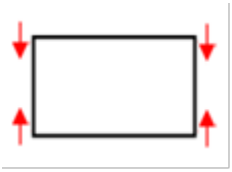
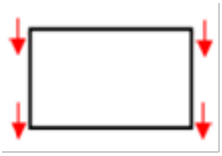
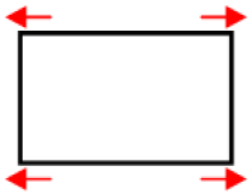
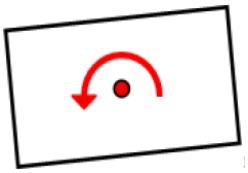
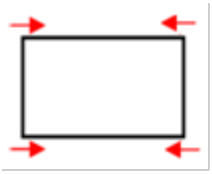
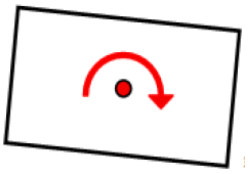
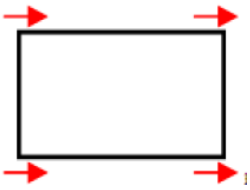
- Parallelogram
- Trapezoid
- Pin balance
- Pincushion
- Shift
- Width
- Height
- Rotation (around bitmap image center point)

For each reference point and corresponding shift vector, the adjustment is computed and applied. The resulting shift vector is again cached in the warping library. This way of implementation allows incremental adjustments.

Following are tables detailing supported adjustments:

Mode Direction	Trapezoid	Parallelogram	Pincushion	Pin balance
Up				
Down				
Left				

Right				

Mode Direction	Height	Width	Rotation	Shift
Up				
Down				
Left				
Right				

Enabling Warping

Following are the display settings required to enable warping in Candera:

1. Enable warping
2. Set warp matrix
3. Set warp image bounds
4. Apply display settings

Candera’s Display class uses a transaction based execution model for setting properties. Several changes can be queued. Changes have to be applied explicitly via a call to Display::ApplyChanges() . Warping matrix data from the warping library can be used directly in Candera.

Warping Sequence

1. Distribute reference points over the bitmap according to the specifications in the parameter list. Parameterized through: Bitmap Area and the associated Bitmap Area Shift
2. Bitmap predistortion (warping) Parameterized through: the active Parameter Set
3. Rotate predistorted image data according to the parameter list. The rotation of the warped reference points must take place around the center point of the warped bitmap. Parameterized through: rotation of the warping area and warped bitmap area center point.
4. Shift rotated and warped image data according to the parameter list. Parameterized through: shift of warped bitmap area center point.
5. Display warped, rotated and shifted image data on the display

Warping Library API

The public interface of the Warping Library is presented below:

Method	Parameters
en_warp_result_t **CalculateWarpMatrix()** • calculates the WarpMatrix from configuration parameters and reference points passed.	• stc_warp_configuration_pamameter_t* [IN] **pstcWarpConfigParams**: pointer to structure holding the configuration parameters and limits that need to be checked. • stc_warp_parameter_set_t* [IN] **pstcWarpRefPoints**: pointer to structure holding the packed reference points. These values are internally copied, so that the caller may change the values in the structure without any side effect to the reference points held by the library. • stc_warp_matrix_t* [OUT] **pstcWarpMatrix**: pointer to data structure in which the WarpMatrix is written to. Memory must be allocated and managed by the caller. > The values written into this data structure are valid only if success code is returned, otherwise data might be inconsistent or incomplete.
en_warp_result_t **AdjustWarpMatrix()** • adjust WarpMatrix from configuration parameters and adjust passed parameters based on internally stored reference points.	• stc_warp_configuration_pamameter_t* [IN] **pstcWarpConfigParams**: pointer to structure holding the configuration parameters and limits that need to be checked. • stc_warp_adjust_parameter_t* [IN] **pstcWarpAdjustParams**: pointer to structure holding the adjustment parameters. • stc_warp_bool_t* [OUT] **pbRefPointsChanged**: pointer to Boolean where the function returns a flag indicating if the internally stored reference points have been changed during the adjustments. Memory must be allocated and managed by the caller. • stc_warp_matrix_t* [OUT] **pstcWarpMatrix**: pointer to data structure in which the WarpMatrix is written to. Memory must be allocated and managed by the caller. > The values written into this data structure are valid only if success code is returned, otherwise data might be inconsistent or incomplete.

en_warp_result_t **GetRefPoints()** • copy the values of internal current and valid reference point to the data structure. **Note:** If any range check fails, appropriate error code is returned.	• stc_warp_parameter_set_t* [OUT] **pstWarpRefPoints**: pointer to data structure where the packed reference points will be copied to. Memory must be allocated and managed by the caller.
en_warp_result_t **ComputeWarpImageBoundsFromConfig()** • computes the bounds of the warping image within the full render target contents in normalized coordindates [0..1] from the warping configuration. **Note:** The bounds can be used as input for SetWarpImageBounds() function of Candera::Display.	• stc_warp_configuration_parameter_t* [IN] **config**: configuration used for warping. • WRP_SFLOAT* [OUT] **x**: coordinate of upper-most left bitmap area point. • WRP_SFLOAT* [OUT] **y**: coordinate of upper-most left bitmap area point • WRP_SFLOAT* [OUT] **width**: width of the bitmap area • WRP_SFLOAT* [OUT] **height**: height of the bitmap area

Warping Candera API

The warping related public interface of the Candera::Display class is presented below:

Methods
void SetWarpMatrix(WarpMatrix &warpMatrix, UInt index=0)
const WarpMatrix & GetWarpMatrix(UInt index=0) const
void ResetWarpMatrix(UInt index)
void SetWarpMatrixWeight(UInt index, Float weight)
Float GetWarpMatrixWeight(UInt index) const
void SetWarpImageBounds(Rectangle &warpImageBounds)
const Rectangle & GetWarpImageBounds() const
void SetWarpOrientation(DisplayOrientation::Enum orientation)
DisplayOrientation::Enum GetWarpOrientation() const
bool SetWarpingEnabled(bool isEnabled)
bool IsWarpingEnabled() const