gsub_lite

Abstract

A collection of useful OpenGL routines for ARToolKit.

Discussion

Sample code for example usage of gsub_lite is included with ARToolKit, in the directory <AR/examples /simpleLite>.

gsub_lite is the preferred means for drawing camera video images acquired from ARToolKit's video libraries. It includes optimized texture handling, and a variety of flexible drawing options.

gsub_lite also provides utility functions for setting the OpenGL viewing frustum and camera position based on ARToolKit- camera parameters and marker positions.

gsub_lite does not depend on GLUT, or indeed, any particular window or event handling system. It is therefore well suited to use in applications which have their own window and event handling code.

gsub_lite v2.7 is intended as a replacement for gsub from ARToolKit 2.65, by Mark Billinghurst (MB) & Hirokazu Kato (HK), with the following additional functionality:

- Support for true stereo and multiple displays through removal of most dependencies on global variables.
- Prepared library for thread-safety by removing global variables.
- Optimised texturing, particularly for Mac OS X platform.
- Added arglCameraFrustum to replace argDraw3dCamera() function.
- Renamed argConvGlpara() to arglCameraView() to more accurately represent its functionality.
- Correctly handle textures with non-RGBA handling.
- Library setup and cleanup functions.
- Version numbering.

It does however lack the following functionality from the original gsub library:

- GLUT event handling.
- Sub-window ("MINIWIN") and half-size drawing.
- HMD support for stereo via stencil.

Functions

---

arglCameraFrustum

Create an OpenGL perspective projection matrix.

```
void arglCameraFrustum(
    const ARParam *cparam,
    const double focalmin,
    const double focalmax,
    GLdouble m_projection[16]);
```

Parameter Descriptions

cparam
Pointer to a set of ARToolKit camera parameters for the current video source.
focalmax
The maximum distance at which geometry will be rendered. Any geometry further away from the camera than this distance will be clipped and will not be appear in a rendered frame. Thus, this value should be set high enough to avoid clipping of any geometry you care about. However, the precision of the depth buffer is correlated with the ratio of focalmin to focalmax, thus you should not set focalmax any higher than it needs to be. This value should be specified in the same units as your OpenGL drawing.
focalmin

The minimum distance at which geometry will be rendered. Any geometry closer to the camera than this distance will be clipped and will not be appear in a rendered frame. Thus, this value should be set low enough to avoid clipping of any geometry you care about. However, the precision of the depth buffer is correlated with the ratio of focalmin to focalmax, thus you should not set focalmin any lower than it needs to be. Additionally, geometry viewed in a stereo projections that is too close to camera is difficult and tiring to view, so if you are rendering stereo perspectives you should set this value no lower than the near-point of the eyes. The near point in humans varies, but usually lies between 0.1 m 0.3 m. This value should be specified in the same units as your OpenGL drawing.

m_projection

Pointer to a array of 16 GLdoubles, which will be filled out with a projection matrix suitable for passing to OpenGL. The matrix is specified in column major order.

Discussion

Use this function to create a matrix suitable for passing to OpenGL to set the viewing projection.

Availability

First appeared in ARToolKit 2.68.

---

arglCameraFrustumRH

(description)

```
void arglCameraFrustumRH(
    const ARParam *cparam,
    const double focalmin,
    const double focalmax,
    GLdouble m_projection[16]);
```

Parameter Descriptions

(name)

(description)

function result

(description)

Discussion

(description)

---

arglCameraView

Create an OpenGL viewing transformation matrix.

```
void arglCameraView(
    const double para[3][4],
    GLdouble m_modelview[16],
    const double scale);
```

Parameter Descriptions

para

Pointer to 3x4 matrix array of doubles which specify the position of an ARToolKit marker, as returned by arGetTransMat().

m_modelview

Pointer to a array of 16 GLdoubles, which will be filled out with a modelview matrix suitable for passing to OpenGL. The matrix is specified in column major order.

scale
Specifies a scaling between ARToolKit's units (usually millimeters) and OpenGL's coordinate system units. What you pass for the scalefactor parameter depends on what units you want to do your OpenGL drawing in. If you use a scalefactor of 1.0, then
To use different OpenGL units, e.g. metres, then you would pass 0.001.

Discussion

Use this function to create a matrix suitable for passing to OpenGL to set the viewing transformation of the virtual camera.

Availability
First appeared in ARToolKit 2.68.

---

arglCameraViewRH

(description)

```
void arglCameraViewRH(
    const double para[3][4],
    GLdouble m_modelview[16],
    const double scale);
```

Parameter Descriptions

(name)
(description)

function result
(description)

Discussion

(description)

---

arglCleanup

Free memory used by gsub_lite associated with the specified context.

```
void arglCleanup(
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Parameter Descriptions

contextSettings
A reference to ARGL's settings for an OpenGL context, as returned by arglSetupForCurrentContext().

Discussion

Should be called after no more argl* functions are needed, in order to prevent memory leaks etc.

The library can be setup again for the context at a later time by calling arglSetupForCurrentContext() again.

Availability

First appeared in ARToolKit 2.68.

---

arglDispImage

Display an ARVideo image, by drawing it using OpenGL.

```
void arglDispImage(
    ARUint8 *image,
    const ARParam *cparam,
    const double zoom,
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Parameter Descriptions

image
: Pointer to the tightly-packed image data (as returned by arVideoGetImage()). The horizontal and vertical dimensions of the image data must exactly match the values specified in the fields cparam->xsize and cparam->ysize (see below).

  The first byte of image data corresponds to the first component of the top-left-most pixel in the image. The data continues with the remaining pixels of the first row, followed immediately by the pixels of the second row, and so on to the last byte of the image data, which corresponds to the last component of the bottom-right-most pixel.

cparam
: Pointer to a set of ARToolKit camera parameters for the current video source. The size of the source image is taken from the fields xsize and ysize of the ARParam structure pointed to. Also, when the draw mode is AR_DRAW_BY_TEXTURE_MAPPING (see the documentation for the global variable arglDrawMode) the field dist_factor of the ARParam structure pointed to will be taken as the amount to un-warp the supplied image.

zoom
: The amount to scale the video image up or down. To draw the video image double size, use a zoom value of 2.0. To draw the video image half size use a zoom value of 0.5.

contextSettings
: A reference to ARGL's settings for the current OpenGL context, as returned by arglSetupForCurrentContext() for this context. It is the callers responsibility to make sure that the current context at the time arglDisplayImage() is called matches that under which contextSettings was created.

Discussion

This function draws an image from an ARVideo source to the current OpenGL context. This operation is most useful in video see-through augmented reality applications for drawing the camera view as a background image, but can also be used in other ways.

An undistorted image is drawn with the lower-left corner of the bottom-left-most pixel at OpenGL screen coordinates (0,0), and the upper-right corner of the top-right-most pixel at OpenGL screen coodinates (x * zoom, y * zoom), where x and y are the values of the fields cparam->xsize and cparam->ysize (see below) and zoom is the value of the parameter zoom (also see below). If cparam->dist_factor indicates that an un-warping correction should be applied, the actual coordinates will differ from the values specified here.

OpenGL state: Drawing is performed with depth testing and lighting disabled, and thus leaves the the depth buffer (if any) unmodified. If pixel transfer is by texturing (see documentation for arglDrawMode), the drawing is done in replacement texture environment mode. The depth test enable and lighting enable state and the texture environment mode are restored before the function returns.

Availability
: First appeared in ARToolKit 2.68.

---

arglDispImageStateful

Display an ARVideo image, by drawing it using OpenGL, using and modifying current OpenGL state.

```
void arglDispImageStateful(
    ARUint8 *image,
    const ARParam *cparam,
    const double zoom,
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Discussion

This function is identical to arglDispImage except that whereas arglDispImage sets an orthographic 2D projection and the OpenGL state prior to drawing, this function does not. It also does not restore any changes made to OpenGL state.

This allows you to do effects with your image, other than just drawing it 2D and with the lower-left corner of the bottom-left-most pixel attached to the bottom-left (0,0) of the window. For example, you might use a perspective projection instead of an orthographic projection with a glLoadIdentity() / glTranslate() on the modelview matrix to place the lower-left corner of the bottom-left-most pixel somewhere other than 0,0 and leave depth-testing enabled.

See the documentation for arglDispImage() for more information.

Availability
First appeared in ARToolKit 2.68.2.

---

arglDistortionCompensationGet

Enquire as to the enable state of camera lens distortion compensation in arglDispImage.

```
int arglDistortionCompensationGet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    int *enable);
```

Parameter Descriptions

`contextSettings`
A reference to ARGL's settings for the current OpenGL context, as returned by arglSetupForCurrentContext() for this context.

`enable`
Pointer to an integer value which will be set to TRUE if distortion compensation is enabled in the specified context, or FALSE if it is disabled.

function result
TRUE if the distortion value was retreived, FALSE if an error occurred.

Discussion

By default, arglDispImage compensates for the distortion of the camera's acquired image caused by the lens when it draws. This function enquires as to whether arglDispImage is currently doing compensation or not.

Availability
First appeared in ARToolKit 2.71.

---

arglDistortionCompensationSet

Set compensation for camera lens distortion in arglDispImage to off or on.

```
int arglDistortionCompensationSet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    int enable);
```

Parameter Descriptions

contextSettings
: A reference to ARGL's settings for the current OpenGL context, as returned by arglSetupForCurrentContext() for this context.

enable
: TRUE to enabled distortion compensation, FALSE to disable it. The default state for new contexts is enable = TRUE.

function result
: TRUE if the distortion value was set, FALSE if an error occurred.

Discussion

By default, arglDispImage compensates for the distortion of the camera's acquired image caused by the lens when it draws. By calling this function with enabled = FALSE, this compensation will be disabled in the specified drawing context. It may be re-enabled at any time. This function is useful if you need to draw an image, but do not know the extent of the camera's lens distortion (such as during distortion calibration). While distortion compensation is disabled, the dist_factor[] array in a the camera cparam structure passed to arglDispImage is ignored.

Availability
: First appeared in ARToolKit 2.71.

---

arglDrawModeGet

Get method by which arglDispImage() is transfering pixels.

```
int arglDrawModeGet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Discussion

Enquires as to the current method by which arglDispImage() is transferring pixels to OpenGL for display. See arglDrawModeSet() for more information.

Availability
: First appeared in ARToolKit 2.72.

---

arglDrawModeSet

Set method by which arglDispImage() will transfer pixels.

```
void arglDrawModeSet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    const int mode);
```

Discussion

This setting determines the method by which arglDispImage transfers pixels of an image to OpenGL for display. Setting this variable to a value of AR_DRAW_BY_GL_DRAW_PIXELS specifies the use of the OpenGL DrawPixels functions to do the transfer. Setting this variable to a value of AR_DRAW_BY_TEXTURE_MAPPING specifies the

use of OpenGL TexImage2D functions to do the transfer. The DrawPixels method is guaranteed to be available on all implementations, but arglDispImage does not correct the image for camera lens distortion under this method. In contrast, TexImage2D is only available on some implementations, but allows arglDispImage() to apply a correction for camera lens distortion, and additionally offers greater potential for accelerated drawing on some implementations.

The initial value is AR_DRAW_BY_TEXTURE_MAPPING.

Availability

First appeared in ARToolKit 2.72.

---

arglPixelFormatGet

Get the format of pixel data in which arglDispImage*() is expecting data to be passed.

```
int arglPixelFormatGet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    AR_PIXEL_FORMAT *format,
    int *size);
```

Parameter Descriptions

contextSettings

A reference to ARGL's settings for the current OpenGL context, as returned by arglSetupForCurrentContext() for this context.

format

A symbolic constant for the pixel format in use. See AR_PIXEL_FORMAT in ar.h for a list of all possible formats.

size

The number of bytes of memory occupied per pixel, for the given format.

function result

TRUE if the pixel format and size values were retreived, FALSE if an error occurred.

Discussion

This function enquires as to the current format of pixel data being expected by the arglDispImage*() functions. The default format is determined by the value of AR_DEFAULT_PIXEL_FORMAT at the time the library was built.

Availability

First appeared in ARToolKit 2.71.

---

arglPixelFormatSet

Set the format of pixel data which will be passed to arglDispImage*()

```
int arglPixelFormatSet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    AR_PIXEL_FORMAT format);
```

Parameter Descriptions

contextSettings

A reference to ARGL's settings for the current OpenGL context, as returned by arglSetupForCurrentContext() for this context.

format

A symbolic constant for the pixel format being set. See AR_PIXEL_FORMAT in ar.h for a list of

all possible formats.

function result
TRUE if the pixel format value was set, FALSE if an error occurred.

Discussion

(description) In gsub_lite, the format of the pixels (i.e. the arrangement of components within each pixel) can be changed at runtime. Use this function to inform gsub_lite the format the pixels being passed to arglDispImage*() functions are in. This setting applies only to the context passed in parameter contextSettings. The default format is determined by the value of AR_DEFAULT_PIXEL_FORMAT at the time the library was built. Usually, image data is passed in directly from images generated by ARVideo, and so you should ensure that ARVideo is generating pixels of the same format.

Availability
First appeared in ARToolKit 2.71.

---

arglSetupForCurrentContext

Initialise the gsub_lite library for the current OpenGL context.

```
ARGL_CONTEXT_SETTINGS_REF arglSetupForCurrentContext(
    void);
```

function result
An ARGL_CONTEXT_SETTINGS_REF. See the documentation for this type for more info.

Discussion

This function performs required setup of the gsub_lite library for the current OpenGL context and must be called before any other argl*() functions are called for this context.

An OpenGL context holds all of the state of the OpenGL machine, including textures and display lists etc. There will usually be one OpenGL context for each window displaying OpenGL content.

Other argl*() functions whose operation depends on OpenGL state will require an ARGL_CONTEXT_SETTINGS_REF. This is just so that they can keep track of per-context variables.

You should call arglCleanup() passing in the ARGL_CONTEXT_SETTINGS_REF when you have finished with the library for this context.

Availability
First appeared in ARToolKit 2.68.

---

arglTexmapModeGet

Enquire whether full or half-resolution TexImage2D pixel-transfer is being used in arglDispImage().

```
int arglTexmapModeGet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Discussion

Enquires as to the current value of the TexmapMode setting. See arglTexmapModeSet() for more info.

Availability
First appeared in ARToolKit 2.72.

---

arglTexmapModeSet

Determines use of full or half-resolution TexImage2D pixel-transfer in arglDispImage().

```
void arglTexmapModeSet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    const int mode);
```

Discussion

When arglDrawModeSet(AR_DRAW_BY_TEXTURE_MAPPING) has been called, the value of this setting determines whether full or half-resolution data is transferred to the texture. Calling this function with a mode value of AR_DRAW_TEXTURE_FULL_IMAGE uses all available pixels in the source image data. A value of AR_DRAW_TEXTURE_HALF_IMAGE discards every second row in the source image data, defining a half-height texture which is then drawn stretched vertically to double its height.

The latter method is well-suited to drawing interlaced images, as would be obtained from DV camera sources in interlaced mode or composite video sources.

The initial value is AR_DRAW_TEXTURE_FULL_IMAGE.

Availability

First appeared in ARToolKit 2.72.

---

arglTexRectangleGet

Enquire as to use of rectangular TexImage2D pixel-transfer in arglDispImage().

```
int arglTexRectangleGet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings);
```

Discussion

Enquires as to the current value of the TexRectangle setting. See arglTexRectangleSet() for more info.

Availability

First appeared in ARToolKit 2.72.

---

arglTexRectangleSet

Determines use of rectangular TexImage2D pixel-transfer in arglDispImage().

```
void arglTexRectangleSet(
    ARGL_CONTEXT_SETTINGS_REF contextSettings,
    const int state);
```

Discussion

On implementations which support the OpenGL extension for rectangular textures (of non power-of-two size), and when arglDrawMode is set to AR_DRAW_BY_TEXTURE_MAPPING, the value of this variable determines whether rectangular textures or ordinary (power-of-two) textures are used by arglDispImage(). A value of TRUE specifies the use of rectangluar textures. A value of FALSE specifies the use of ordinary textures.

If the OpenGL driver available at runtime does not support for rectangular textures, changing the value of this setting to TRUE will result calls to arglDispImage performing no drawing.

Availability
First appeared in ARToolKit 2.72.

Typedefs

ARGL_CONTEXT_SETTINGS_REF

Opaque type to hold ARGL settings for a given OpenGL context.

```
typedef struct _ARGL_CONTEXT_SETTINGS *ARGL_CONTEXT_SETTINGS_REF;
```

Discussion

An OpenGL context is an implementation-defined structure which keeps track of OpenGL state, including textures and display lists. Typically, individual OpenGL windows will have distinct OpenGL contexts assigned to them by the host operating system.

As gsub_lite uses textures and display lists, it must be able to track which OpenGL context a given texture or display list it is using belongs to. This is especially important when gsub_lite is being used to draw into more than one window (and therefore more than one context.)

Basically, functions which depend on OpenGL state, will require an ARGL_CONTEXT_SETTINGS_REF to be passed to them. An ARGL_CONTEXT_SETTINGS_REF is generated by setting the current OpenGL context (e.g. if using GLUT, you might call glutSetWindow()) and then calling arglSetupForCurrentContext(). When you have finished using ARGL in a given context, you should call arglCleanup(), passing in an ARGL_CONTEXT_SETTINGS_REF, to free the memory used by the settings structure.

Availability
First appeared in ARToolKit 2.68.