gsub.h File Reference

ARToolkit OpenGL subroutines. More...

```
#include <AR/config.h>
#include <AR/param.h>
#include <AR/ar.h>
```

Functions

void	**argInit** (**ARParam** *cparam, double zoom, int fullFlag, int xwin, int ywin, int hmd_flag) *Initialise the gsub library.*
void	**argLoadHMDparam** (**ARParam** *lparam, **ARParam** *rparam) *initialize camera for HMD.*
void	**argCleanup** (void) *Close the gsub library.*
void	**argSwapBuffers** (void) *swap the rendering buffer.*
void	**argMainLoop** (void(*mouseFunc)(int button, int state, int x, int y), void(*keyFunc)(unsigned char key, int x, int y), void(*mainFunc)(void)) *start the program main loop with specified callback functions.*
void	**argDrawMode2D** (void) *switch the rendering context for 2D rendering mode.*
void	**argDraw2dLeft** (void) *switch the rendering view to left eye (in 2D space)*
void	**argDraw2dRight** (void) *switch the rendering view to right eye (in 2D space)*
void	**argDrawMode3D** (void) *switch the rendering context for 3D rendering mode.*
void	**argDraw3dLeft** (void) *switch the rendering view to left eye (in 3D space)*
void	**argDraw3dRight** (void) *switch the rendering view to right eye (in 3D space)*
void	**argDraw3dCamera** (int xwin, int ywin) *switch the rendering view for 3D rendering mode.*
void	**argConvGlpara** (double para[3][4], double gl_para[16]) *transform ARToolKit matrix format to an openGL matrix format.*
void	**argConvGLcpara** (**ARParam** *param, double gnear, double gfar, double m[16]) *transform ARToolKit intrinsic camera parameters matrix format to an openGL matrix format.*
void	**argDispImage** (ARUint8 *image, int xwin, int ywin) *display the video image.*
void	**argDispHalfImage** (ARUint8 *image, int xwin, int ywin) *display half of the video image.*
void	**argDrawSquare** (double vertex[4][2], int xwin, int ywin) *draw a 2D square.*
void	**argLineSeg** (double x1, double y1, double x2, double y2, int xwin, int ywin) *Draw a line.*
void	**argLineSegHMD** (double x1, double y1, double x2, double y2) *Draw a line with HMD mode.*
void	**argInqSetting** (int *hmdMode, int *gMiniXnum2, int *gMiniYnum2, void(**mouseFunc)(int button, int state, int x, int y), void(**keyFunc)(unsigned char key, int x, int y), void(**mainFunc)(void)) *Get current configuration of gsub library.*
void	**argsInit** (ARSParam *scparam, double zoom, int twinFlag, int fullFlag, int xwin, int ywin)
void	**argsDraw3dCamera** (int xwin, int ywin, int LorR, int stencil_flag)
void	**argsConvGLcpara** (ARSParam *sparam, double gnear, double gfar, double mL[16], double mR[16])
void	**argsDispImage** (ARUint8 *image, int LorR, int xwin, int ywin)
void	**argsDispHalfImage** (ARUint8 *image, int LorR, int xwin, int ywin)
void	**argsLineSeg** (double x1, double y1, double x2, double y2, int xwin, int ywin, int LorR)
void	**argsDrawSquare** (double vertex[4][2], int xwin, int ywin, int LorR)

Variables

int	**argDrawMode** *define the draw configuration mode.*
int	**argTexmapMode** *define the texture map configuration mode.*

Detailed Description

ARToolkit OpenGL subroutines.

This file contains the main display functions used in ARToolkit Library. It contains wrapped functions of GLUT and openGL for doing real-time rendering. This choice give us a large multi-platform support for the display module of ARToolkit.

Remarks:

The supported stereo mode is interleaved stereo mode (only for i-glasses display).

History :

Author:
Hirokazu Kato kato@sys.im.hiroshima-cu.ac.jp

Version:

Date:

Function Documentation

```
void argCleanup( void )
```

Close the gsub library.

This function clean the rendering context (GLUT and openGL). Call in the exit of your program.

Remarks:
BE CAREFUL, THIS FUNCTION DOESN'T RESET PERSPECTIVE MATRIX AND CURRENT GL STATE TO DEFAULT

```
void argConvGLcpara( ARParam * param,
                     double    gnear,
                     double    gfar,
                     double    m[16]
                   )
```

transform ARToolKit intrinsic camera parameters matrix format to an openGL matrix format.

XXXBK: not be sure of this function: this function must just convert 3x4 matrix to classical perspective openGL matrix. But in the code, you used arParamDecompMat that seem decomposed K and R,t, aren't it ? why do this decomposition since we want just intrinsic parameters ? and if not what is arDecomp ?

Returned value is generally use with a Projection Matrix.

Parameters:

param	
gnear	near clipping plane value
gfar	far clipping plane value
m	the resulted openGL matrix

```
void argConvGlpara( double para[3][4],
                    double gl_para[16]
                  )
```

transform ARToolKit matrix format to an openGL matrix format.

simple conversion for the openGL matrix (16 values and homogeneous matrix). Returned value is generally use with a Model View Matrix.

Parameters:

para	the ARToolKit matrix
gl_para	the resulted openGL matrix

```
void argDispHalfImage( ARUint8 * image,
                       int       xwin,
                       int       ywin
                     )
```

display half of the video image.

Idem of argDispImage except than a quarter of the image is display (first left top quadrant, so size/2 in x and y).

Parameters:

image	image to display
xwin	XXXBK
ywin	XXXBK

```
void argDispImage( ARUint8 * image,
                   int       xwin,
                   int       ywin
                 )
```

display the video image.

Display in the back-buffer the video image in argument. For doing AR video background, this function must be called before any rendering of 3D object.

Remarks:
According to your argDrawMode, argTexmapMode and the internal image format the openGL function called is different and less or more efficient.

with AR_DRAW_BY_GL_DRAW_PIXELS, unaffected by current camera parameters matrix but affected by glRasterPos3f.

with AR_DRAW_BY_TEXTURE_MAPPING, affected by current current camera parameters matrix. You need generally call argDrawMode2D before this function.

Parameters:

image	image to display
xwin	XXXBK
ywin	XXXBK

```
void argDraw2dLeft( void )
```

switch the rendering view to left eye (in 2D space)

Combine with argDrawMode2D for rendering the left view.

```
void argDraw2dRight( void )
```

switch the rendering view to right eye (in 2D space)

Combine with argDrawMode2D for rendering the right view.

```
void argDraw3dCamera( int xwin,
                      int ywin
                    )
```

switch the rendering view for 3D rendering mode.

Update curent internal camera parameters for rendering in 3D space. this function complements argDrawMode3D.

Parameters:

xwin	length of rendering view (less than window length)
ywin	width of rendering view (less than window width)

```
void argDraw3dLeft( void )
```

switch the rendering view to left eye (in 3D space)

Update curent internal camera parameters for rendering in 3D space for left eye. this function complements argDrawMode3D.

```
void argDraw3dRight( void )
```

switch the rendering view to right eye (in 3D space)

Update curent internal camera parameters for rendering in 3D space for left eye. this function complements argDrawMode3D.

```
void argDrawMode2D( void )
```

switch the rendering context for 2D rendering mode.

Update curent camera parameters (internal and external) for rendering 2D or 3D objects in the view plane (like text or 2D shape). This function define an orthographic projection in the image plane. It not define opengl state for rendering in image space (like for a bitmap copy).

```
void argDrawMode3D( void )
```

switch the rendering context for 3D rendering mode.

Update curent camera parameters for rendering in 3D space. Generally call to reinializing model view matrix.

```
void argDrawSquare( double vertex[4][2],
                    int    xwin,
                    int    ywin
                  )
```

draw a 2D square.

Draw a square. The position of the square is affected by openGL model view matrix and call to argDrawMode2D argDrawMode3D. Generally call in a 2D mode (so after a argDrawMode2D).

Parameters:

vertex	corner of square.
xwin	XXXBK
ywin	XXXBK

```
argInit( ARParam * cparam,
         double    zoom,
         int       fullFlag,
         int       xwin,
         int       ywin,
         int       hmd_flag
       )
```

Initialise the gsub library.

This function performs required initialisation of the gsub library. It must be called before any other argl*() functions are called.

Parameters:

cparam	the intrinsics parameters of the camera (used to defined openGL perspective matrix)

zoom	defined a zoom parameter for the final result.
fullFlag	full screen mode (1 enable, 0 disable).
xwin	XXXBK. 0 if indifferent.
ywin	XXXBK. 0 if indifferent.
hmd_flag	enable stereo display mode (only interleaved configuration)

```
argInqSetting( int *                                         hmdMode,
               int *                                         gMiniXnum2,
               int *                                         gMiniYnum2,
               void(**)(int button, int state, int x, int y) mouseFunc,
               void(**)(unsigned char key, int x, int y)     keyFunc,
               void(**)(void)                                mainFunc
             )
```

Get current configuration of gsub library.

Retrieve current state of gsub library like the current callback functions.

Parameters:

hmdMode	the current hmdMode
gMiniXnum2	XXXBK
gMiniYnum2	XXXBK
mouseFunc	the current mouse function callback
keyFunc	the current key function callback
mainFunc	the current main function callback

```
void argLineSeg( double x1,
                 double y1,
                 double x2,
                 double y2,
                 int    xwin,
                 int    ywin
               )
```

Draw a line.

Draw a segment.T The position of the line is affected by openGL model view matrix and call to argDrawMode2D argDrawMode3D. Generally call in a 2D mode (so after a argDrawMode2D).

Parameters:

x1	x position of the first point.
y1	y position of the first point.
x2	x position of the second point.
y2	y position of the second point.
xwin	XXXBK
ywin	XXXBK

```
void argLineSegHMD( double x1,
                    double y1,
                    double x2,
                    double y2
                  )
```

Draw a line with HMD mode.

Draw a segment in HMD mode.

Parameters:

x1	x position of the first point.
y1	y position of the first point.
x2	x position of the second point.
y2	y position of the second point.

```
void argLoadHMDparam( ARParam * lparam,
                      ARParam * rparam
                    )
```

initialize camera for HMD.

Load in the display module the intrinsic parameters of the two view, i.e camera (identify to the eyes).

Parameters:

lparam	parameter of left camera
rparam	parameter of right camera

```
void argMainLoop( void(*)(int button, int state, int x, int y) mouseFunc,
                  void(*)(unsigned char key, int x, int y)     keyFunc,
                  void(*)(void)                                mainFunc
                )
```

start the program main loop with specified callback functions.

This function is called in the entry block of a program. User specify the main callback of his program. Users should not put routines calls after this function, generally never accessible.

Parameters:

mouseFunc	the user mouse function can be NULL.
keyFunc	the user keyboard function can be NULL.
mainFunc	the user main update function can be NULL.

void argSwapBuffers(void)

swap the rendering buffer.

Swap the back-buffer to the front-buffer. the pre-condition is that all the rendering functions have been called.

Variable Documentation

int argDrawMode

define the draw configuration mode.

Define the draw mode for display of the video background. The possible values are :

- AR_DRAW_BY_GL_DRAW_PIXELS: use the GL_DRAW_PIXELS function
- AR_DRAW_BY_TEXTURE_MAPPING: use a quad mesh with a texture mapping of the video. by default: DEFAULT_DRAW_MODE in config.h choice and performance depends on your hardware and your openGL driver.

int argTexmapMode

define the texture map configuration mode.

If the draw mode is AR_DRAW_BY_TEXTURE_MAPPING, you can configure the copy mode of the texture mapping. The possible values are :

- AR_DRAW_TEXTURE_FULL_IMAGE: texture mapping full resolution.
- AR_DRAW_TEXTURE_HALF_IMAGE: texture mapping half resolution. by default: DEFAULT_DRAW_TEXTURE_IMAGE in config.h