Warp3D

COLLABORATORS

	TITLE : Warp3D	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 16, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Warp3D

1.1 Warp3D

W3D_AllocStencilBuffer()

W3D_AllocTexObj()

W3D_AllocZBuffer()

W3D_BestModeID()

W3D_CheckDriver()

W3D_CheckIdle()

W3D_ClearDrawRegion()

W3D_ClearStencilBuffer()

W3D_ClearZBuffer()

W3D_CreateContext()

W3D_DestroyContext()

W3D_DrawLine()

W3D_DrawLineLoop()

W3D_DrawLineStrip()

W3D_DrawPoint()

W3D_DrawTriangle()

W3D_DrawTriangleV()

W3D_DrawTriFan()

W3D_DrawTriFanV()

W3D_DrawTriStrip()

W3D_DrawTriStripV()

W3D_FillStencilBuffer()

W3D_Flush()

W3D_FlushFrame()

W3D_FlushTextures()

W3D_FreeAllTexObj()

W3D_FreeScreenmodeList()

W3D_FreeStencilBuffer()

W3D_FreeTexObj()

W3D_FreeZBuffer()

W3D_GetDestFmt()

W3D_GetDrivers()

W3D_GetDriverState()

W3D_GetDriverTexFmtInfo()

W3D_GetScreenmodeList()

W3D_GetState()

W3D_GetTexFmtInfo()

W3D_Hint()

W3D_LockHardware()

W3D_Query()

W3D_QueryDriver()

W3D_ReadStencilPixel()

W3D_ReadStencilSpan()

W3D_ReadZPixel()

W3D_ReadZSpan()

W3D_ReleaseTexture()

W3D_RequestMode()

W3D_SetAlphaMode()

W3D_SetBlendMode()

W3D_SetChromaTestBounds()

W3D_SetColorMask()

W3D_SetCurrentColor()

W3D_SetCurrentPen()

W3D_SetDrawRegion()

W3D_SetDrawRegionWBM()

W3D_SetFilter()

W3D_SetFogParams()

W3D_SetLogicOp()

W3D_SetPenMask()

W3D_SetScissor()

W3D_SetState()

W3D_SetStencilFunc()

W3D_SetStencilOp()

W3D_SetTexEnv()

W3D_SetWrapMode()

W3D_SetWriteMask()

W3D_SetZCompareMode()

W3D_TestMode()

W3D_UnLockHardware()

W3D_UpdateTexImage()

W3D_UpdateTexSubImage()

W3D_UploadTexture()

W3D_WaitIdle()

W3D_WriteStencilPixel()

W3D_WriteStencilSpan()

W3D_WriteZPixel()

```
                W3D_WriteZSpan()
```

1.2 Warp3D/W3D_AllocStencilBuffer()

```
                  NAME
  W3D_AllocStencilBuffer -- Allocate stencil buffer

SYNOPSIS
  success =  W3D_AllocStencilBuffer(context);
  d0                                a0

  ULONG W3D_AllocStencilBuffer(W3D_Context *);

FUNCTION
  Allocate a stencil buffer for the given context. For more
  information on stencil buffering, see the OpenGL specs.

INPUTS
  context - The context the stencil buffer is allocated on

RESULT
  One of the following values:
    W3D_SUCCESS         The allocation was successful
    W3D_NOGFXMEM        No memory was left on the graphics board
    W3D_NOSTENCILBUFFER Stencil buffering is not available

EXAMPLE

NOTES
  Stencil buffering and the ViRGE: The ViRGE is not capable of stencil
  buffering, it became a necessity later when hardware accelerators
  started to support the OpenGL standard.

BUGS

SEE ALSO

             W3D_FreeStencilBuffer
```

1.3 Warp3D/W3D_AllocTexObj()

```
                  NAME
  W3D_AllocTexObj -- Allocate a new texture object

SYNOPSIS
  texture = W3D_AllocTexObj(context, error, ATOTags);
  d0                        a0       a1     a2

  W3D_Texture *W3D_AllocTexObj(W3D_Context, ULONG *, struct TagItem *);
```

```
FUNCTION
  Create a new texture object. Such a texture object contains
  information about a texture in addition to the normal image data
  that is displayed.

INPUTS
  context - pointer to a W3D_Context
  error   - pointer to a ULONG, which will contain an error code,
        or NULL if you do not want to get the error code.
  ATOTags - pointer to a taglist. Supported tags are:
        W3D_ATO_IMAGE (mandatory):
          A pointer to the source texture image
        W3D_ATO_FORMAT (mandatory):
          The texture format of the source texture. Must be
          one of the following values (check the include file
          for more precise definition):
          - W3D_CHUNKY
          - W3D_A1R5G5B5
          - W3D_R5G6B5
          - W3D_R8G8B8
          - W3D_A4R4G4B4
          - W3D_A8R8G8B8
          - W3D_R8G8B8A8
          - W3D_A8
          - W3D_L8
          - W3D_L8A8
          - W3D_I8
        W3D_ATO_WIDTH (mandatory):
          The width of the texture in pixels. Must
          be 2^n.
        W3D_ATO_HEIGHT (mandatory):
          The height of the texture in pixels. Must
          be 2^n.
        W3D_ATO_MIPMAP (optional):
          If specified, the texture can be used for mipmapping.
          The value of this tag defines, which mipmap levels
          have to be generated automatically. It should be set
          so that the generated mipmaps and the provided ones
          build a complete mipmap set.
          The value is a bitmask with one specific bit
          representing a mipmap level. Bit 0 corresponds to
          level 1, Bit 1 to level 2, so Bit n to level n-1.
          A value of 0 means, that all mipmaps are provided
          by the application.
          Note, that providing only a part of all mipmaps
          which leave holes between the provided levels may
          result in performance loss.
        W3D_ATO_MIPMAPPTRS (mandatory for user-supplied mipmaps)
          If W3D_ATO_MIPMAP is specified, mipmapping is used
          for texturing. The mipmap mask specifies which of the
          mipmaps will be created. With the W3D_ATO_MIPMAPPTRS tag,
          an array of (void *) to the mipmaps you want to
          supply yourself is defined. This array must be
          NULL-Terminated
          Example: You want to give only level 3 and 5, and
          let W3D_AllocTexObj create the rest of the mipmaps.
          Assume a 128x128 texture (7 mipmap levels)
```

```
          Define an array like this:
            void *mips[3];
            mips[0] = (void *)level_3_map;
            mips[1] = (void *)level_5_map;
            mips[2] = NULL;
          When calling W3D_AllocTexObj, you would give
          W3D_ATO_MIPMAP the value 0x6B (binary 1101011)
          W3D_ATO_MIPMAPPTRS would be mips.
        W3D_ATO_PALETTE (mandatory for chunky textures):
          Defines the palette which is necessary to handle
          chunky textures. A pointer to a palette must be
          provided. The palette itself is an array of
          ULONG's, and every ULONG defines the ARGB value
          for one color index. Therefore the palette must
          be 1024 bytes. (Note: On 8bit screens, this
          palette *should* be the screen palette,
          unless the driver returns TRUE on W3D_Q_PALETTECONV.)

RESULT
  Either a pointer to the successfully created texture
  object, or NULL, in which case the optional error variable
  is set to one of the following values:
    W3D_SUCCESS             It worked!
    W3D_ILLEGALINPUT        Some information was invalid, maybe
                a mandatory tag missing
    W3D_NOMEMORY            No memory was available
    W3D_UNSOPPORTEDTEXSIZE  The driver can't handle a texture
                of the given size.
    W3D_NOPALETTE           The texture should be a chunky (CLUT)
                texture, but no palette was given.
    W3D_UNSUPPORTEDTEXFMT   The format can not be used with the
                current driver

EXAMPLE
  extern W3D_Context *context;
  void *image = LoadImage("texture.iff");
  W3D_Texture *texobj;
  struct TagItem tags[] = {
    W3D_ATO_IMAGE,      image,
    W3D_ATO_FORMAT,     W3D_A1R5G5B5,
    W3D_ATO_WITDH,      128,
    W3D_ATO_HEIGHT,     128,
    TAG_DONE,           0
  };
  ULONG error;

  texobj = W3D_AllocTexObj(context, &error, tags);
  if (!texobj)
    printf("An error has occurred because: An error has occurred (%d)\n",
          error);


NOTES
  The pointers to textures and mipmaps passed to this function are
  considered `locked` until this texture object is released again,
  or the image is updated with
            W3D_UpdateTexImage
```

```
            .
  You may not free the memory.

BUGS

SEE ALSO

              W3D_FreeTexObj
              ,
              W3D_ReleaseTexture
              ,
              W3D_UpdateTexImage
              ,

              W3D_FlushTextures
              ,
              W3D_SetFilter
              ,
              W3D_SetTexEnv
              ,
              W3D_SetWrapMode

              W3D_UploadTexture
```

1.4 Warp3D/W3D_AllocZBuffer()

```
                NAME
  W3D_AllocZBuffer -- Allocate a ZBuffer

SYNOPSIS
  result = W3D_AllocZBuffer(context);
  d0                        a0

  ULONG W3D_AllocZBuffer(W3D_Context *);

FUNCTION
  Allocates a ZBuffer. The size of the ZBuffer depends on the
  size of the bitmap used with this context. The memory is allocated
  on the graphics board.

INPUTS
  context - pointer to the context to be used with the ZBuffer

RESULT
  One of the following values:
    W3D_SUCCESS     The allocation was successful
    W3D_NOGFXMEM    Not enough video memory
    W3D_NOZBUFFER   ZBuffering is not available on this hardware
    W3D_NOTVISIBLE  - The bitmap is not visible/swapped out of vmem

EXAMPLE
  ULONG error, status;
  struct BitMap myBitMap;
  struct TagItem taglist[] = {
```

```
    W3D_CC_BITMAP,       (ULONG)&myBitMap,
    W3D_CC_YOFFSET,      0,
    W3D_CC_DRIVERTYPE,   W3D_DRIVER_BEST
  };
  W3D_Context *context;

  InitBitMap(&myBitMap, 15, 640, 480);
  createPlanes(&myBitMap);
  context =
            W3D_CreateContext
            (&error, taglist);
  status = W3D_AllocZBuffer(context);

NOTES
  This function should be called before textures are uploaded to
  the graphics board, to avoid fragmentation of video memory.

BUGS

SEE ALSO

             W3D_FreeZBuffer
```

1.5 Warp3D/W3D_BestModeID()

```
NAME
  W3D_BestModeID -- Find a suitable ModeID (V3)

SYNOPSIS
  ModeID = W3D_BestModeID(tags);
  ModeID = W3D_BestModeIDTags(Tag1, ...);

  ULONG W3D_BestModeID(struct TagItem *tags);
  ULONG W3D_BestModeIDTags(Tag tag1, ...);

FUNCTION
  Returns a screen mode ID that best fits the parameters
  supplied in the tag list.

INPUTS
  tags - a taglist, consisting of the following possible tag item:
    W3D_BMI_DRIVER          Must work with this driver
    W3D_BMI_WIDTH           Must have approximately this width
    W3D_BMI_HEIGHT          Must have approximately this height
    W3D_BMI_DEPTH           Must have at least this depth

RESULT
  ModeID - A screenmode ID or INVALID_ID in case of error

EXAMPLE

NOTES

BUGS
```

```
SEE ALSO
```

1.6 Warp3D/W3D_CheckDriver()

```
NAME
  W3D_CheckDriver -- Check driver availability

SYNOPSIS
  flags = W3D_CheckDriver();
  d0

  ULONG W3D_CheckDriver(void);

FUNCTION
  Checks what driver is available (CPU/HW), and returns it
  as a bit mask.

INPUTS
  None

RESULT
  A long word that has it's bit set accordingly:
    W3D_DRIVER_3DHW - A hardware driver is available
    W3D_DRIVER_CPU  - A software driver is available

EXAMPLE
  ULONG flags = W3D_CheckDriver();
  if (flags & W3D_DRIVER_3DHW) printf("Hardware driver available\n");
  if (flags & W3D_DRIVER_CPU)  printf("Software driver available\n");

NOTES
  This function can be called without a valid context. It can
  be used to evaluate the possibilities the system is offering.
  Note though, that you should give the user a chance to get into
  your program, even if you think it would be too slow without
  hardware acceleration...

BUGS

SEE ALSO
```

1.7 Warp3D/W3D_CheckIdle()

```
               NAME
  W3D_CheckIdle -- check if hardware is working

SYNOPSIS
  working = W3D_CheckIdle(context);
  d0                      a0

  ULONG W3D_CheckIdle(W3D_Context *);
```

```
FUNCTION
  Check if the hardware is finished with it's current operation.

INPUTS
  context - a pointer to a W3D_Context

RESULT
  One of to values indicating busy/idle state:
    W3D_SUCCESS - The hardware is idle
    W3D_BUSY    - The hardware is still working

EXAMPLE

NOTES
  This function is not very useful for applications.

BUGS

SEE ALSO

             W3D_WaitIdle
```

1.8 Warp3D/W3D_ClearDrawRegion()

```
NAME
  W3D_ClearDrawRegion -- Clear the current drawing area

SYNOPSIS
  success = W3D_ClearDrawRegion(context, color);
  d0                            a0       d0

  ULONG W3D_ClearDrawRegion(W3D_Context *, ULONG);

FUNCTION
  ClearDrawRegion clears the current drawing area to the color
  given by color. The operation may performed with the boards
  blitter, so this is the prefered way for clearing. Additionally,
  this call makes using V39 multibuffering easier by prociding
  a way to clear the back buffer.

INPUTS
  context - A pointer to the context to use
  color   - The color value to clear to. For direct color drawing
      regions (i.e. TrueColor/HiColor), this is a 32 bit
      color value in the form ARGB, with each component 8 bit.
      For 8 bit (palettized) screens, it's an 8 bit color
      index. Note that for the first form, the color is
      always 8 bits per component, regardless of the color
      format of the drawing region (15/16/24/32 bit).

RESULT
  One of the following:
    W3D_SUCCESS         The operation was successful
```

```
    W3D_NOTVISIBLE      The contex was not in locked state

EXAMPLE

NOTES
  W3D_NOTVISIBLE is a bit misleading. The drawing area must not
  be visible on the screen to be clear, but the context must be
  locked, so that the screen buffer is in memory

BUGS

SEE ALSO
```

1.9 Warp3D/W3D_ClearStencilBuffer()

```
                NAME
  W3D_ClearStencilBuffer -- Clear the stencil buffer

SYNOPSIS
  success = W3D_ClearStencilBuffer(context, clearval);
  d0                               a0       a1

  ULONG W3D_ClearStencilBuffer(W3D_Context *, ULONG *);

FUNCTION
  Clear the stencil buffer (fill it up) with the value
  pointed to by clearval.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context  - the context to work on
  clearval - pointer to a value used for clearing

RESULT
  One of the following:
    W3D_SUCCESS         Operation was successful
    W3D_NOSTENCILBUFFER Stencil buffer not present (not allocated,
                or not supported by driver)
    W3D_NOTVISIBLE      The stencil buffer can not be accessed by
                the hardware
    W3D_QUEUEFAILED     In indirect mode only. Queueing this request
                failed

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_AllocStencilBuffer
             ,
             W3D_FreeStencilBuffer
```

1.10 Warp3D/W3D_ClearZBuffer()

```
                NAME
  W3D_ClearZBuffer -- Clear the ZBuffer with a given value

SYNOPSIS
  success = W3D_ClearZBuffer(context, clearvalue);
  d0                         a0       a1

  ULONG W3D_ClearZBuffer(W3D_Context *, W3D_Double *);

FUNCTION
  Clear the ZBuffer with a given value.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context     - pointer to the context
  clearvalue  - pointer to a W3D_Double, ranging from [0..1].
        If NULL, 0.0 is used

RESULT
  One of the following values:
    W3D_SUCCESS     operation successful
    W3D_NOZBUFFER   No ZBuffer was allocated
    W3D_NOTVISIBLE  The ZBuffer was not in video ram
    W3D_QUEUEFAILED     In indirect mode only. Queueing this request
                failed

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_AllocZBuffer
             ,
             W3D_FreeZBuffer
```

1.11 Warp3D/W3D_CreateContext()

```
                NAME
  W3D_CreateContext -- Create a new Warp3D context

SYNOPSIS
  context = W3D_CreateContext(&error, CCTags);
  D0                          A0      A1
```

```
  W3D_Context *W3D_CreateContext(ULONG *, struct TagItem *);

FUNCTION
  This function creates a new Warp3D context, which is required by most
  other API functions as first parameter.

  The number of open contexts is not limited. Full multitasking capabilities
  are provided.

INPUTS
  error   - A pointer to a ULONG which gets the error value,
        or NULL if you don't want an error code returned
  CCTags  - A taglist containing various input parameters:
    W3D_CC_MODEID (special):
      Specifies the ModeID of the screen you opened or
      intend to open, or generally the ModeID of the drawing
      area you intend to use. If you plan to use Warp3D in
      windowed mode, you may leave this tag unset. Otherwise,
      the tag MUST be set correctly, as the ModeID is used to
      extract the required hardware.
    W3D_CC_BITMAP (mandatory):
      A pointer to the bitmap which is used for 3D drawing.
      For 3DHW drivers, the bitmap must absolutely be located
      in video memory (it may be swapped out at the moment).
      For CPU drivers, it doesn't matter, where the bitmap is
      located. Note, that CPU drivers might use FAST-RAM
      buffers for intermediate results to speed up rendering,
      therefore bitmaps in FAST-RAM might not be optimal in
      this case.
      Also note, that never bitmaps should be provided which
      are directly visible!
    W3D_CC_YOFFSET (mandatory):
      A vertical offset, which defines, at which Y-Position
      the drawing area starts. This can be used to achieve
      multibuffering using the ScrollVPort trick, which might
      be the only possibility to achieve proper multibuffering
      with some graphics interface software.
    W3D_CC_DRIVERTYPE (mandatory):
      A constant which defines what type of driver should
      be used (use the API function
             W3D_CheckDriver
              to get
      more information about the drivers). Possible values
      are:
      - W3D_DRIVER_BEST       the best driver is chosen
      - W3D_DRIVER_3DHW       the hardware driver is chosen,
                  if none is present, NULL is
                  returned
      - W3D_DRIVER_CPU        the software driver is chosen,
                  if none is present, NULL is
                  returned
    W3D_CC_W3DBM (optional):
      Boolean tag. If this is set to TRUE, the W3D_CC_BITMAP
      tag doesn't point to a struct BitMap. Instead, it points
      to a Warp3D bitmap (of type W3D_Bitmap), which might
      be in fast-ram (for CPU rendering). Note that the
```

```
      W3D_CC_YOFFSET tag is ignored if W3D_CC_W3DBM is set
      to TRUE.
    W3D_CC_INDIRECT (optional):
      Boolean tag. If set to TRUE, then all drawing actions are
      possibly not performed directly, but are queued until
      the buffer is full, or
             W3D_Flush
              is called, or the
      indirect state is switched off with
             W3D_SetState
                    W3D_CC_GLOBALTEXENV (optional):
      Boolean tag. If set to TRUE, calls to SetTexEnv do not
      modify the given texture, but are used for all textures.
    W3D_CC_DOUBLEHEIGHT (optional):
      Boolean tag. This tag should be set to TRUE if the drawing
      area is a double height screen. Double height screens
      may be used for double buffering with CyberGraphX.
    W3D_CC_FAST: (optional):
      Boolean tag. If set to TRUE, drawing functions are allowed
      to modify the passed structures.


RESULT
  A pointer to a newly created context structure, or NULL for failure.
  If an error variable was provided, the error value is filled in.
  It may be one of the following values:
    W3D_SUCCESS         - Operation was successful
    W3D_ILLEGALINPUT    - Illegal input, maybe a left out tag item
    W3D_NOMEMORY        - Unable to get enough memory
    W3D_NODRIVER        - No driver was available
    W3D_UNSUPPORTEDFMT  - The supplied bitmap can't be supported
    W3D_ILLEGALBITMAP   - The bitmap is not properly initialised

EXAMPLE
  ULONG error;
  struct BitMap myBitMap;
  struct TagItem taglist[] = {
    W3D_CC_BITMAP,      (ULONG)&myBitMap,
    W3D_CC_YOFFSET,     0,
    W3D_CC_DRIVERTYPE,  W3D_DRIVER_BEST
  };
  W3D_Context *context;

  InitBitMap(&myBitMap, 15, 640, 480);
  createPlanes(&myBitMap);
  context = W3D_CreateContext(&error, taglist);

NOTES
  An error of type W3D_UNSUPPORTEDFMT is returned if a W3D_Bitmap
  is given as drawregion and no CPU driver is available, or
  a HW driver is also requested.

BUGS

SEE ALSO

             W3D_DestroyContext
```

```
              ,
              W3D_Flush
              ,
              W3D_SetState
```

1.12 Warp3D/W3D_DestroyContext()

```
                 NAME
  W3D_DestoryContext -- Release a Warp3D context

SYNOPSIS
  W3D_DestoryContext(context);
           A0

  void W3D_DestroyContext(W3D_Context *);

FUNCTION
  This function frees up all resources for the given context,
  destroying it.

INPUTS
  context - Pointer to a Warp3D context

RESULT
  None

EXAMPLE
  W3D_Context *context;
  ...
  context =
             W3D_CreateContext
             (.....);
  ...
  W3D_DestroyContext(context);

NOTES
  Always release contexts. Even if the memory loss doesn't kill you,
  the hardware may be blocked.

BUGS

SEE ALSO

              W3D_CreateContext
```

1.13 Warp3D/W3D_DrawLine()

```
NAME
  W3D_DrawLine -- Draw a three-dimensional line
```

```
SYNOPSIS
  success = W3D_DrawLine(context, line);
  d0                     a0       a1

  ULONG W3D_DrawLine(W3D_Context *, W3D_Line *);

FUNCTION
  This function draws a line based on the current state.
  It may only be used while the hardware is locked, except when
  indirect drawing is used.

INPUTS
  context - The context to be drawn in
  line    - Definition of a line.

RESULT
  A value inidcating success or failure. One of the following:
    W3D_SUCCESS         (you guessed it!)
    W3D_NOTEXTURE       The line has no texture
    W3D_TEXNOTRESIDENT  The required texture is not in video ram
    W3D_NOGFXMEM        No memory available on the graphics card
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES
  The linewidth parameter will probably not be supported
  by most 3D hardware.

BUGS

SEE ALSO
```

1.14 Warp3D/W3D_DrawLineLoop()

```
                NAME
  W3D_DrawLineLoop -- Draw a closed sequence of connected lines (V2)

SYNOPSIS
  success = W3D_DrawLineLoop(context, lines);
  d0                         a0       a1

  ULONG W3D_DrawLineLoop(W3D_Context *, W3D_Lines *);

FUNCTION
  This function draws a connected sequence of lines, similar to
  the
             W3D_DrawLineStrip
              function. The only difference is that the
  last vertex is connected to the first with a line segment, too,
  meaning that the vertexcount lines are drawn.

INPUTS
```

```
  context     - pointer to the context.
  lines       - pointer to the W3D_Lines (not the trailing 's')
        structure defining the line strip.

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Fewer than two vertices were given
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES

BUGS
  Currently, this call is not queued.

SEE ALSO
  W3D_DrawLineLoop,
             W3D_DrawLine
```

1.15 Warp3D/W3D_DrawLineStrip()

```
                NAME
  W3D_DrawLineStrip -- Draw a sequence of connected lines (V2)

SYNOPSIS
  success = W3D_DrawLineStrip(context, lines);
  d0                          a0       a1

  ULONG W3D_DrawLineStrip(W3D_Context *, W3D_Lines *);

FUNCTION
  Draws a sequence of connected lines (a line strip). The first
  line is defined by vertices 0 and 1, the second line by vertices
  1 and 2, ..., up to the last line being defined by vertices
  n-1 and n, with n being the vertexcount field from the W3D_Lines
  structure.

INPUTS
  context     - pointer to the context.
  lines       - pointer to the W3D_Lines (not the trailing 's')
        structure defining the line strip.

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
```

```
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Fewer than two vertices were given
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES

BUGS
  Currently, this call is not queued.

SEE ALSO

             W3D_DrawLineLoop
             ,
             W3D_DrawLine
```

1.16 Warp3D/W3D_DrawPoint()

```
NAME
  W3D_DrawPoint -- Draw a point

SYNOPSIS
  success = W3D_DrawPoint(context, point);
  d0                      a0       a1

  ULONG W3D_DrawPoint(W3D_Context *, W3D_Point *);

FUNCTION
  Draw a point based on the current context
  It may only be used while the hardware is locked, except when
  indirect drawing is used.

INPUTS
  context - a pointer to the context to draw with
  point   - a pointer to a filled W3D_Point

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board`s memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES
  The pointsize parameter will probably not be supported by most
  3D hardware.

  Although the vertex has it's own color, the GOURAUD shading state
  must be enabled to use this color, otherwise the current color set
```

```
  by W3D_SetCurrentColor/W3D_SetCurrentPen will be used.

BUGS

SEE ALSO
```

1.17 Warp3D/W3D_DrawTriangle()

```
                NAME
  W3D_DrawTriangle -- Draw a triangle

SYNOPSIS
  success = W3D_DrawTriangle(context, triangle);
  d0                         a0       a1

  ULONG W3D_DrawTriangle(W3D_Context *, W3D_Triangle *);

FUNCTION
  Draw a triangle to the given context, based on that context's
  state.
  It may only be used while the hardware is locked, except when
  indirect drawing is used.

INPUTS
  context     - the context to be drawn to
  triangle    - the triangle to be drawn

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_DrawTriFan
             ,
             W3D_DrawTriStrip
```

1.18 Warp3D/W3D_DrawTriangleV()

```
                NAME
  W3D_DrawTriangleV -- Draw a triangle

SYNOPSIS
  success = W3D_DrawTriangleV(context, triangle);
  d0                          a0       a1

  ULONG W3D_DrawTriangleV(W3D_Context *, W3D_TriangleV *);

FUNCTION
  Draw a triangle to the given context, based on that context's
  state.
  It may only be used while the hardware is locked.
  Indirect drawing is not supported by this call.
  This is the "vectorized" version; instead of inlined vertex
  structures, it uses pointers.

INPUTS
  context     - the context to be drawn to
  triangle    - the triangle to be drawn

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES
  Requires Warp3D V3

BUGS

SEE ALSO

             W3D_DrawTriFanV
             ,
             W3D_DrawTriStripV
```

1.19 Warp3D/W3D_DrawTriFan()

```
                NAME
  W3D_DrawTriFan -- Draw a triangle fan

SYNOPSIS
  success = W3D_DrawTriFan(context, triangles);
  d0                       a0       a1

  ULONG W3D_DrawTriFan(W3D_Context *, W3D_Triangles *);
```

```
FUNCTION
  Draw a triangle fan. The first vertex in the list is
  considered the common point for the fan. For more
  information on triangle fans, see the OpenGL specs.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context     - pointer to the context.
  triangles   - pointer to a vertex list. Note that this
        is a W3D_Triangles (trailing s, avoid mixing
        up with W3D_Traingle)

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Less than three vertices were given
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_DrawTriangle
             ,
             W3D_DrawTriStrip
```

1.20 Warp3D/W3D_DrawTriFanV()

```
                NAME
  W3D_DrawTriFanV -- Draw a triangle fan

SYNOPSIS
  success = W3D_DrawTriFanV(context, triangles);
  d0                        a0       a1

  ULONG W3D_DrawTriFanV(W3D_Context *, W3D_TrianglesV *);

FUNCTION
  Draw a triangle fan. The first vertex in the list is
  considered the common point for the fan. For more
  information on triangle fans, see the OpenGL specs.
  This function may only be used while the hardware is locked.
  Indirect drawing is not supported by this call.
  This is the "vectorized" version. Instead of suplying a
```

```
  pointer to an array of vertex structure, you supply a pointer
  to an array of vertex structure pointers.

INPUTS
  context     - pointer to the context.
  triangles   - pointer to a vertex list. Note that this
        is a W3D_TrianglesV (trailing s, avoid mixing
        up with W3D_TraingleV)

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Less than three vertices were given
    W3D_QUEUEFAILED     The request can't be queued in indirect mode
    W3D_NOMEMORY        The feature should have been emulated since the
              driver does not support it, but memory alloc failed.

EXAMPLE

NOTES
  Requires Warp3D V3

BUGS

SEE ALSO

             W3D_DrawTriangleV
             ,
             W3D_DrawTriStripV
```

1.21 Warp3D/W3D_DrawTriStrip()

```
                NAME
  W3D_DrawTriStrip -- Draw a triangle strip

SYNOPSIS
  success = W3D_DrawTriStrip(context, triangles);
  d0                         a0       a1

  ULONG W3D_DrawTriStrip(W3D_Context *, W3D_Triangles *);

FUNCTION
  Draw a triangle strip. For more information
  on triangle strips, see the OpenGL specs.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context     - pointer to the context.
  triangles   - pointer to a vertex list. Note that this
```

```
        is a W3D_Triangles (trailing s, avoid mixing
        up with W3D_Traingle)

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board's memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Less than three vertices were given
    W3D_QUEUEFAILED     The request can't be queued in indirect mode

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_DrawTriangle
             ,
             W3D_DrawTriFan
```

1.22 Warp3D/W3D_DrawTriStripV()

```
                NAME
  W3D_DrawTriStripV -- Draw a triangle strip

SYNOPSIS
  success = W3D_DrawTriStripV(context, triangles);
  d0                        a0       a1

  ULONG W3D_DrawTriStripV(W3D_Context *, W3D_TrianglesV *);

FUNCTION
  Draw a triangle strip. For more information
  on triangle strips, see the OpenGL specs.
  This function may only be used while the hardware is locked.
  Indirect drawing is not supported for this function.
  This is the "vectorized" version. Instead of suplying a
  pointer to an array of vertex structure, you supply a pointer
  to an array of vertex structure pointers.

INPUTS
  context     - pointer to the context.
  triangles   - pointer to a vertex list. Note that this
        is a W3D_Triangles (trailing s, avoid mixing
        up with W3D_Traingle)

RESULT
  One of the following:
    W3D_SUCCESS         It worked.
```

```
    W3D_NOTEXTURE       No texture given
    W3D_TEXNOTRESIDENT  The texture is not on the graphics board‘s memory
    W3D_NOTVISIBLE      The drawing area is not visible
    W3D_NOZBUFFER       No ZBuffer present, although it has been requested
    W3D_ILLEGALINPUT    Less than three vertices were given
    W3D_QUEUEFAILED     The request can’t be queued in indirect mode

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_DrawTriangle
             ,
             W3D_DrawTriFan
```

1.23 Warp3D/W3D_FillStencilBuffer()

```
                NAME
  W3D_FillStencilBuffer -- Fill the stencil buffer

SYNOPSIS
  success = W3D_FillStencilBuffer(context, x, y, width, height, depth, data);
  d0                              a0       d0 d1 d2     d3      d4     a1

  ULONG W3D_FillStencilBuffer(W3D_Context *, ULONG, ULONG, ULONG, ULONG,
           ULONG, void *);

FUNCTION
  This function fills the stencil buffer with a rectangular image
  with the given dimensions.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - the context
  x,y     - Coordinates into the stencil buffer
  width   - Width of the image data
  height  - Height of the image data
  depth   - Depth of the image data. Must be 8,16 or 32
  data    - The data itself

RESULT
  One of the following values:
    W3D_SUCCESS             Operation successful
    W3D_NOSTENCILBUFFER     No stencil buffer present (either it‘s not
                  allocated, or not supported)
    W3D_ILLEGALINPUT        Illegal depth value
    W3D_NOTVISIBLE          The stencil buffer can not be accessed by
                  the hardware
```

```
EXAMPLE

NOTES

BUGS

SEE ALSO
  W3D_CreateStencilBuffer,
             W3D_ClearStencilBuffer
```

1.24 Warp3D/W3D_Flush()

```
                NAME
  W3D_Flush -- Flush indirect drawing queue

SYNOPSIS
  result = W3D_Flush(context);
      a0

  ULONG W3D_Flush(W3D_Context *);

FUNCTION
  If the given context is not in indirect mode, nothing happens.
  Otherwise, the internal queue is flushed and all buffered drawing
  request are drawn.

INPUTS
  context - the context which should be flushed

RESULT
  A value indicating error or success:
    W3D_SUCCESS     success
    W3D_NOTVISIBLE  Locking the hardware was unsuccesful

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_SetState
             ,
             W3D_CreateContext
             ,
             W3D_LockHardware
             ,
             W3D_UnLockHardware
```

1.25 Warp3D/W3D_FlushFrame()

```
NAME
  W3D_FlushFrame -- Flush the current frame

SYNOPSIS
  W3D_FlushFrame(context);
          a0

  void W3D_FlushFrame(W3D_Context*);

FUNCTION
  This function flushes the current frame. It must be called at the end
  of your drawing when the frame is finished. This function *must* be
  called by any application, even if you do not "intent" to support
  CPU drivers (for which this function is mainly designed).

INPUTS
  context -   The context to flush

RESULT

EXAMPLE

NOTES
  If the context is indirect, this function also flushes the
  Queue.

BUGS

SEE ALSO
```

1.26 Warp3D/W3D_FlushTextures()

```
                NAME
  W3D_FlushTextures -- Release all textures from video ram

SYNOPSIS
  W3D_FlushTextures(context);
          a0

  void W3D_FlushTextures(W3D_Context);

FUNCTION
  This function releases every texture that`s currently
  on the graphics board`s texture memory.

INPUTS
  context - Pointer to a W3D_Context

RESULT
  None

EXAMPLE
```

```
NOTES

BUGS

SEE ALSO

             W3D_ReleaseTexture
```

1.27 Warp3D/W3D_FreeAllTexObj()

```
                NAME
  W3D_FreeAllTexObj -- Free all textures in context

SYNOPSIS
  W3D_FreeAllTexObj(context);
          a0

  void
             W3D_FreeTexObj
             (W3D_Context *);

FUNCTION
  Free all texture objects allocated in the current context.

INPUTS
  context - the pointer to the context

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_FreeTexObj
             ,
             W3D_AllocTexObj
```

1.28 Warp3D/W3D_FreeScreenmodeList()

```
                NAME
  W3D_FreeScreenmodeList - Free the list of screen modes (V3)

SYNOPSIS
  void W3D_FreeScreenmodeList(W3D_ScreenMode *);
```

```
  W3D_FreeScreenmodeList(list);

FUNCTION
  Frees all resources that are attached to the Screen Mode list
  which must have been allocated with
             W3D_GetScreenmodeList
             .

INPUTS
  list - the list pointer obtained by
             W3D_GetScreenmodeList
                RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_GetScreenmodeList
```

1.29 Warp3D/W3D_FreeStencilBuffer()

```
NAME
  W3D_FreeStencilBuffer -- Free the stencil buffer

SYNOPSIS
  success = W3D_FreeStencilBuffer(context);
  d0                              a0

  ULONG W3D_FreeStencilBuffer(W3D_Context *);

FUNCTION
  Free up all memory associated with the stencil buffer.

INPUTS
  context - the context containing the stencil buffer to be freed

RESULT
  One of the following values:
    W3D_SUCCESS             Operation succesful
    W3D_NOSTENCILBUFFER     No stencil buffer was allocated, or stencil
                  buffering is not supported by the current
                  hardware driver.
    W3D_NOTVISIBLE          The stencil buffer can not be accessed by
                  the hardware

EXAMPLE

NOTES

BUGS
```

```
SEE ALSO
  W3D_CreateStencilBuffer
```

1.30 Warp3D/W3D_FreeTexObj()

```
                NAME
  W3D_FreeTexObj -- Free a texture object

SYNOPSIS
  W3D_FreeTexObj(context, texture);
         a0       a1

  void W3D_FreeTexObj(W3D_Context *, W3D_Texture *);

FUNCTION
  Remove the texture object from the list of textures
  and free up all resources associated with it.

INPUTS
  context - Pointer to a W3D_Context
  texture - Pointer to a texture to be released

RESULT
  None

EXAMPLE
  extern W3D_Context *context;
  void *image = LoadImage("texture.iff");
  W3D_Texture *texobj;
  struct TagItem tags[] = {
    W3D_ATO_IMAGE,      image,
    W3D_ATO_FORMAT,     W3D_A1R5G5B5,
    W3D_ATO_WITDH,      128,
    W3D_ATO_HEIGHT,     128,
    TAG_DONE,           0
  };
  ULONG error;

  texobj =
             W3D_AllocTexObj
             (context, &error, tags);
  if (!texobj) {
    printf("An error has occurred because: An error has occurred (%d)\n",
           error);
  } else {
    ... Draw some cool stuff ...
    W3D_FreeTexObj(context, texobj);


NOTES
  Free all textures. Even if you can afford the memory loss in main memory,
  you'll loose video memory.
  The 'locked' pointers (those to the image and user-defined mipmaps)
  are now 'unlocked', and may be used again.
```

```
BUGS

SEE ALSO

             W3D_AllocTexObj
```

1.31 Warp3D/W3D_FreeZBuffer()

```
                NAME
  W3D_FreeZBuffer -- Free ZBuffer

SYNOPSIS
  success = W3D_FreeZBuffer(context);
  d0                        a0

  ULONG W3D_FreeZBuffer(W3D_Context *);

FUNCTION
  Free the ZBuffer previously allocated with
            W3D_AllocZBuffer
               INPUTS
  context - Pointer to a W3D_Context

RESULT
  One of the following values:
    W3D_SUCCESS     Success
    W3D_NOZBUFFER   No Z Buffer was allocated
    W3D_NOTVISIBLE  ZBuffer is not visible

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_AllocZBuffer
```

1.32 Warp3D/W3D_GetDestFmt()

```
                NAME
  W3D_GetDestFmt -- Get information about supported formats

SYNOPSIS
  format = W3D_GetDestFmt();
  d0

  ULONG W3D_GetDestFmt(void);
```

```
FUNCTION
  *DEPRECATED* DO NOT USE THIS IN NEW PROJECTS
  This function can be used to get information about the destination
  (i.e. screen) format supported by the current driver. The result
  is a bitmask, with each bit representing a supported format.
  This function can be used before opening a display, to ensure
  that only a supported display area is selected.

INPUTS
  None

RESULT
  A bitmask representing supported modes. Currently, some of the
  following bits:
    W3D_FMT_CLUT
    W3D_FMT_R5G5B5
    W3D_FMT_B5G5R5
    W3D_FMT_R5G5B5PC
    W3D_FMT_B5G5R5PC
    W3D_FMT_R5G6B5
    W3D_FMT_B5G6R5
    W3D_FMT_R5G6B5PC
    W3D_FMT_B5G6R5PC
    W3D_FMT_R8G8B8
    W3D_FMT_B8G8R8
    W3D_FMT_A8R8G8B8
    W3D_FMT_A8B8G8R8
    W3D_FMT_R8G8B8A8
    W3D_FMT_B8G8R8A8

EXAMPLE
  ULONG fmt = W3D_GetDestFmt();

  if (fmt & W3D_FMT_CLUT)     printf("Driver supports 8 bit modes\n");
  if (fmt & W3D_R5G5B5)       printf("Driver supports 15 bit RGB modes\n");

NOTES
  This function is deprecated and should not be used in future
  projects.

BUGS

SEE ALSO

              W3D_CreateContext
              ,
              W3D_Query
              ,
              W3D_GetDrivers
```

1.33 Warp3D/W3D_GetDrivers()

```
                NAME
  W3D_GetDrivers -- Get the internal list of drivers (V2)
```

```
SYNOPSIS
  driverarray = W3D_GetDrivers();
   D0

  W3D_Driver **W3D_GetDrivers(void);

FUNCTION
  This function returns a (NULL-Terminated) Array of pointers
  to W3D_Driver structures. You can use these to find a suitable
  driver, offer the user a selection of hardware, or activate
  one driver for further queries.

INPUTS

RESULT
  driverarray -   A null-terminated array of pointers to
          W3D_Driver structures.

EXAMPLE

NOTES
  The returned list is STRICTLY read-only.

BUGS

SEE ALSO

              W3D_TestMode
```

1.34 Warp3D/W3D_GetDriverState()

```
                NAME
  W3D_GetDriverState -- get current state of driver

SYNOPSIS
  result = W3D_GetDriverState(context);
  d0                          a0

  ULONG W3D_GetDriverState(W3D_Context *);

FUNCTION
  Return information about the current state of the driver.
  This function can be used to check if the current driver
  is able to start rendering now.

INPUTS
  context - The context to check the state for

RESULT
  One of the following values:
    W3D_SUCCESS         Success, rendering possible
    W3D_NOTVISIBLE      Drawing area is not currently on
              the video card's memory.
```

```
EXAMPLE
  if (W3D_SUCCESS == W3D_GetDriverState(context)
    RenderFrame();
  else
    printf("Error: Bitmap not visible, can't render\n");

NOTES

BUGS

SEE ALSO

             W3D_LockHardware
```

1.35 Warp3D/W3D_GetDriverTexFmtInfo()

```
                NAME
  W3D_GetDriverTexFmtInfo -- Get information about the texture format (V2)

SYNOPSIS
  info = W3D_GetDriverTexFmtInfo(driver, format, destfmt);
  d0                              a0       d0      d1

  ULONG W3D_GetDriverTexInfo(W3D_Driver*, ULONG, ULONG);

FUNCTION
  This function is used to get information about the texture
  format, i.e. if it's directly supported by the hardware,
  or must be converted in some way. Contrary to the similar
  function W3d_GetTexFmtInfo, this function does not need a
  context to operate, but can be used to query individual drivers
  about their texture format capabilities.

INPUTS
  driver  - A pointer to a W3D_Driver structure
  texfmt  - The texture format to be queried. Currently,
      one of the following:
        W3D_CHUNKY           palettized
        W3D_A1R5G5B5         a rrrrr ggggg bbbbb
        W3D_R5G6B5           rrrrr gggggg bbbbb
        W3D_R8G8B8           rrrrrrrr gggggggg bbbbbbbb
        W3D_A4R4G4B4         aaaa rrrr gggg bbbb
        W3D_A8R8G8B8         aaaaaaaa rrrrrrrr gggggggg bbbbbbbb
        W3D_R8G8B8A8         rrrrrrrr gggggggg bbbbbbbb aaaaaaaa
        W3D_A8               aaaaaaaa
        W3D_L8               llllllll
        W3D_L8A8             llllllll aaaaaaaa
        W3D_I8               iiiiiiii
      See the main documentation for more information.
  destfmt - The destination screen format.

RESULT
  A bitvector with the following bits
```

```
    W3D_TEXFMT_FAST         Format directly supported by HW
    W3D_TEXFMT_CLUTFAST     Format directly supported in CLUT modes only
    W3D_TEXFMT_ARGBFAST     Format directly supported in direct color
                  modes only
    W3D_TEXFMT_UNSUPPORTED  Format not supported, and can't be emulated
    W3D_TEXFMT_SUPPORTED    Format is supported, although it may be
                  internally converted

EXAMPLE

NOTES
  Formats that are not directly supported can still be used for textures.
  Note, however, that those textures must be converted.

BUGS

SEE ALSO

             W3D_GetTexFmtInfo()
```

1.36 Warp3D/W3D_GetScreenmodeList()

```
                NAME
  W3D_GetScreenmodeList - Return a list of screen modes (V3)

SYNOPSIS
  W3D_ScreenMode *W3D_GetScreenmodeList(void)

  list = W3D_GetScreenmodeList();

FUNCTION
  Returns a list of W3D_ScreenMode structures that represent
  all modes that are usable by Warp3D's drivers.

  The result is read-only, step through the list by examining
  the 'Next' field until this is NULL.

INPUTS
  None

RESULT
  list - A pointer to the first W3D_ScreenMode entry or NULL
     if no screenmode was found

EXAMPLE

NOTES
  This function also returns screenmodes which are only usable
  by software drivers. You should examine the Driver field to
  find a mode that matches your desired driver.

  You MUST free this list with
             W3D_FreeScreenmodeList
                BUGS
```

```
   SEE ALSO

               W3D_FreeScreenmodeList
```

1.37 Warp3D/W3D_GetState()

```
                   NAME
     W3D_GetState -- Get current state of hardware/context

   SYNOPSIS
     result = W3D_GetState(context, state);
     d0                    a0       d0

     ULONG W3D_GetState(W3D_Context *, ULONG);

   FUNCTION
     This function reads the state of the bits in the
     state field of the context structure.

   INPUTS
     context - pointer to a Warp3D context
     state   - The bit that is tested. Currently, this may
         be one of the following:
           W3D_AUTOTEXMANAGEMENT  automatic texture management
           W3D_SYNCHRON           wait, until HW is idle
           W3D_INDIRECT           buffer drawings until
                W3D_Flush()
                'ed
           W3D_GLOBALTEXENV       global texture modes
           W3D_DOUBLEHEIGHT       screen has double height.
           W3D_FAST               Drawing functions may modify passed struc
tures
           W3D_TEXMAPPING         texmapping state
           W3D_PERSPECTIVE        perspective correction state
           W3D_GOURAUD            gouraud/flat shading
           W3D_ZBUFFER            Z-Buffer state
           W3D_ZBUFFERUPDATE      Z-Buffer update state
           W3D_BLENDING           Alpha blending state
           W3D_FOGGING            Fogging state
           W3D_ANTI_POINT         Point antialiasing
           W3D_ANTI_LINE          Line antialiasing
           W3D_ANTI_POLYGON       Polygon antialiasing
           W3D_ANTI_FULLSCREEN    Fullscreen antialiasing
           W3D_DITHERING          dithering state
           W3D_LOGICOP            logical operations state
           W3D_STENCILBUFFER      stencil buffer state
           W3D_ALPHATEST          Alpha test state
           W3D_SPECULAR           Specular highlightung state
           W3D_TEXMAPPING3D       3D texturemapping state
              W3D_CHROMATEST        Chroma test (color keying)

   RESULT
     One of the following:
       W3D_ENABLED     the mode is enabled
```

```
    W3D_DISABLED    the mode is disabled/not available

EXAMPLE
  if (W3D_ENABLED == W3D_GetState(context, W3D_FOGGING)) {
    printf("Gee, I can't see in all this fog\n");
  } else {
    printf("Aha, that's better\n");
  }

NOTES
 Don't use W3D_SYNCHRON, this state might only be useful for
 debugging purposes.

 The W3D_FAST mode can speed up your application, always use it,
 if you don't care what happens to the values in the drawing
 structures (like W3D_Triangle, W3D_Line etc.)

 'Indirect drawing' has the advantage, that the 'locking' time
 is minimized, please provide at least an option for the user to
 use it.

 For more information about the different states, please refer
 to the Warp3D Programmer Documentation.

BUGS

SEE ALSO

             W3D_SetState
```

1.38 Warp3D/W3D_GetTexFmtInfo()

```
                NAME
  W3D_GetTexFmtInfo -- Get information about the texture format

SYNOPSIS
  info = W3D_GetTexFmtInfo(context, format, destfmt);
  d0                       a0       d0      d1

  ULONG W3D_GetTexInfo(W3D_Context, ULONG, ULONG);

FUNCTION
  This function is used to get information about the texture
  format, i.e. if it's directly supported by the hardware,
  or must be converted in some way.

INPUTS
  context - A valid context pointer
  texfmt  - The texture format to be queried. Currently,
      one of the following:
        W3D_CHUNKY           palettized
        W3D_A1R5G5B5         a rrrrr ggggg bbbbb
        W3D_R5G6B5           rrrrr gggggg bbbbb
        W3D_R8G8B8           rrrrrrrr gggggggg bbbbbbbb
```

```
        W3D_A4R4G4B4         aaaa rrrr gggg bbbb
        W3D_A8R8G8B8         aaaaaaaa rrrrrrrr gggggggg bbbbbbbb
        W3D_R8G8B8A8         rrrrrrrr gggggggg bbbbbbbb aaaaaaaa
        W3D_A8               aaaaaaaa
        W3D_L8               llllllll
        W3D_L8A8             llllllll aaaaaaaa
        W3D_I8               iiiiiiii
      See the main documentation for more information.
  destfmt - The destination screen format.

RESULT
  A bitvector with the following bits
    W3D_TEXFMT_FAST         Format directly supported by HW
    W3D_TEXFMT_CLUTFAST     Format directly supported in CLUT modes only
    W3D_TEXFMT_ARGBFAST     Format directly supported in direct color
                  modes only
    W3D_TEXFMT_UNSUPPORTED  Format not supported, and can't be emulated
    W3D_TEXFMT_SUPPORTED    Format is supported, although it may be
                  internally converted

EXAMPLE
  ULONG info = W3D_GetTexFmtInfo(NULL, W3D_CHUNKY, W3D_FMT_CLUT);
  if (info & W3D_TEXFMT_CLUTFAST) printf("Supported in CLUT modes\n");

NOTES
  Formats that are not directly supported can still be used for textures.
  Note, however, that those textures must be converted.

  IMPORTANT: Prior to Version 2 of the API, this function could be
  called with a NULL context to query the default driver. Although this
  is still possible for backward compatibility reasons, a programmer
  must not use this feature in new projects, but rather use the new and
  improved
            W3D_GetDriverTexFmtInfo()
             function instead, which is essential
  for multiple driver support. You may still call this function with a
  valid context, of course.

BUGS

SEE ALSO

            W3D_GetDriverTexFmtInfo()
```

1.39 Warp3D/W3D_Hint()

```
                NAME
  W3D_Hint -- Hint about rendering quality

SYNOPSIS
  result = W3D_Hint(context, mode, quality);
  d0                a0       d0    d1

  ULONG W3D_Hint(W3D_Context, ULONG, ULONG);
```

```
FUNCTION
  Gives Warp3D a hint about the desired quality of some
  effects. This can be used to improve rendering speed
  at the cost of display quality.

INPUTS
  context - The context to hint for
  mode    - The mode to hint for. One of the following values
        W3D_H_TEXMAPPING      - quality of general texmapping
        W3D_H_MIPMAPPING      - quality of mipmapping
        W3D_H_BILINEARFILTER  - quality of bilinear filtering
        W3D_H_MMFILTER        - quality of depth filter
        W3D_H_PERSPECTIVE     - quality of perspective correction
        W3D_H_BLENDING        - quality of alpha blending
        W3D_H_FOGGING         - quality of fogging
        W3D_H_ANTIALIASING    - quality of antialiasing
        W3D_H_DITHERING       - quality of dithering
        W3D_H_ZBUFFER         - quality of ZBuffering
  quality - The desired quality. Possible  values are
        W3D_H_FAST            - fast, low quality
        W3D_H_AVERAGE         - average speed, average quality
        W3D_H_NICE            - low speed, high quality


RESULT
  A value indicating success or failure:
    W3D_SUCCESS         Success
    W3D_ILLEGALINPUT    Failure, illegal input

EXAMPLE

NOTES
  This function only gives hints to Warp3D. It is possible
  that it doesn't do anything at all, depending on the
  possibility the hardware or driver offers.

BUGS
  The ViRGE driver selects it's filter modes when they are set
  with
             W3D_SetFilter
             , so you have to set the filter modes again
  when messing with the W3D_H_BILINEARFILTER setting.

SEE ALSO
```

1.40 Warp3D/W3D_LockHardware()

```
                NAME
  W3D_LockHardware -- Gain exclusive hardware access

SYNOPSIS
  res = W3D_LockHardware(context);
  d0                     a0
```

```
  ULONG W3D_LockHardware(W3D_Context *);

FUNCTION
  This function gains exclusive access to the hardware. It must be
  called whenever objects are drawn, except when operating in 'indirect
  render' mode. You should not lock the frame too long, because the
  system is freezed in locked state.

INPUTS
  context - a pointer to a W3D_Context structure

RESULT
  A value indication success or failure:
    W3D_SUCCESS     - The hardware is locked
    W3D_NOTVISIBLE  - The bitmap is not visible/swapped out of vmem

EXAMPLE
  if (W3D_SUCCESS == W3D_LockHardware(context) {
    ...
    Render some stuff
    ...

              W3D_UnLockHardware
              (context);
  } else {
    printf("Can't lock hardware\n");
  }

NOTES
  This function may forbid multitasking (depending on the driver),
  or even disable interrupts.

BUGS

SEE ALSO

              W3D_UnLockHardware
              ,
              W3D_SetState
```

1.41 Warp3D/W3D_Query()

```
                NAME
  W3D_Query -- Query capabilities of the driver

SYNOPSIS
  res = W3D_Query(context, query, destfmt)
  d0              a0       d0     d1

  ULONG W3D_Query(W3D_Context *, ULONG, ULONG);

FUNCTION
  This function is used to query the hardware/driver
```

```
  capabilities. It takes destination formats into account
  (checking compatibility).

INPUTS
  context     - pointer to a W3D_Context
  query       - a value to be queried.
        Currently, the following values are supported:
          W3D_Q_DRAW_POINT         point drawing
          W3D_Q_DRAW_LINE          line drawing
          W3D_Q_DRAW_TRIANGLE      triangle drawing
          W3D_Q_DRAW_POINT_X       points with size != 1 supported
          W3D_Q_DRAW_LINE_X        lines with width != 1 supported
          W3D_Q_DRAW_LINE_ST       line stippling supported
          W3D_Q_DRAW_POLY_ST       polygon stippling supported
          W3D_Q_TEXMAPPING         texmapping in general
          W3D_Q_MIPMAPPING         mipmapping
          W3D_Q_BILINEARFILTER     bilinear filter
          W3D_Q_MMFILTER           mipmap filter
          W3D_Q_LINEAR_REPEAT      W3D_REPEAT for linear texmapping
          W3D_Q_LINEAR_CLAMP       W3D_CLAMP for linear texmapping
          W3D_Q_PERPESCTIVE        perspective correction
          W3D_Q_PERSP_REPEAT       W3D_REPEAT for persp. texmapping
          W3D_Q_PERSP_CLAMP        W3D_CLAMP for persp. texmapping
          W3D_Q_ENV_REPLACE        texenv REPLACE
          W3D_Q_ENV_DECAL          texenv DECAL
          W3D_Q_ENV_MODULATE       texenv MODULATE
          W3D_Q_ENV_BLEND          texenv BLEND
          W3D_Q_FLATSHADING        flat shading
          W3D_Q_GOURAUDSHADING     gouraud shading
          W3D_Q_ZBUFFER            Z buffer in general
          W3D_Q_ZBUFFERUPDATE      Z buffer update
          W3D_Q_ZCOMPAREMODES      Z buffer compare modes
          W3D_Q_ALPHATEST          alpha test in general
          W3D_Q_ALPHATESTMODES     alpha test modes
          W3D_Q_BLENDING           alpha blending
          W3D_Q_SRCFACTORS         source factors
          W3D_Q_DESTFACTORS        destination factors
          W3D_Q_FOGGING            fogging in general
          W3D_Q_LINEAR             linear fogging
          W3D_Q_EXPONENTIAL        exponential fogging
          W3D_Q_S_EXPONENTIAL      square exponential fogging
          W3D_Q_ANTIALIASING       antialiasing in general
          W3D_Q_ANTI_POINT         point antialiasing
          W3D_Q_ANTI_LINE          line antialiasing
          W3D_Q_ANTI_POLYGON       polygon antialiasing
          W3D_Q_ANTI_FULLSCREEN    fullscreen antialiasing
          W3D_Q_DITHERING          dithering
          W3D_Q_SCISSOR            scissor test
          W3D_Q_MAXTEXWIDTH        max. texture width
          W3D_Q_MAXTEXHEIGHT       max. texture height
          W3D_Q_RECTTEXTURES       rectangular textures
          W3D_Q_LOGICOP            logical operations
          W3D_Q_MASKING            color/index masking
          W3D_Q_STENCILBUFFER      stencil buffer in general
          W3D_Q_STENCIL_MASK       mask value
          W3D_Q_STENCIL_FUNC       stencil functions
          W3D_Q_STENCIL_SFAIL      stencil operation SFAIL
```

```
              W3D_Q_STENCIL_DPFAIL     stencil operation DPFAIL
              W3D_Q_STENCIL_DPPASS     stencil operation DPPASS
              W3D_Q_STENCIL_WRMASK     stencil buffer supports write masking
              W3D_Q_PALETTECONV        driver can use texture with a pallett
e
                           other than the screen palette on
                           8 bit screens
              W3D_Q_DRAW_POINT_FX      driver supports point fx (fog, zbuffer
)
              W3D_Q_DRAW_POINT_TEX     driver supports points textured
              W3D_Q_DRAW_LINE_FX       driver supports line fx
              W3D_Q_DRAW_LINE_TEX      driver supports textured lines
              W3D_Q_SPECULAR           driver supports specular reflection
      destfmt     - The destination format

    RESULT
      Depends on the item. With most of the "is this supported"-type
      queries, one of the following constants is returned:
        W3D_FULLY_SUPPORTED     Completely supported by driver
        W3D_PARTIALLY_SUPPORTED Only partially supported
        W3D_NOT_SUPPORTED       Not supported

      With "what is the value"-type queries like W3D_Q_MAXTEXWIDTH,
      an ULONG is returned.

    EXAMPLE
      switch(W3D_Query(context, W3D_Q_TEXMAPING, destfmt)) {
      case W3D_FULLY_SUPPORTED:     printf("Completely supported by driver\n");
                    break;
      case W3D_PARTIALLY_SUPPORTED: printf("Only partially supported\n");
                    break;
      case W3D_NOT_SUPPORTED:       printf("Not supported\n");
                    break;
      }

    NOTES
      Regarding chunky/ARGB combinations:
      You are advised that you always use chunky textures with chunky
      screens only, and ARGB textures with ARGB screens

      IMPORTANT: Prior to Version 2 of the API, the W3D_Query function
      could be called with a NULL pointer instead of a context. Although
      this possibility is still supported for backward compatibility,
      the programmer is strictly encouraged to use the new
                 W3D_QueryDriver
                      function instead. The
                 W3D_QueryDriver
                       function may be used to directly
      query a specific driver for capabilities, which is essential when
      working with V2+ and multiple drivers.

    BUGS

    SEE ALSO

                 W3D_QueryDriver()
```

1.42 Warp3D/W3D_QueryDriver()

```
                NAME
  W3D_QueryDriver -- Query capabilities of any driver (V2)

SYNOPSIS
  res = W3D_QueryDriver(driver, query, destfmt)
  d0                    a0       d0     d1

  ULONG W3D_QueryDriver(W3D_Driver *, ULONG, ULONG);

FUNCTION
  This function is similar to the
            W3D_Query
             function, only
  that it does not require a context but rather operates on
  a driver obtained by
            W3D_GetDrivers()
            .

INPUTS
  driver  -   A pointer to a W3D_Driver structure obtained by

            W3D_GetDrivers()
                 query   -   The data item to be queried. See
            W3D_Query()
             for
       a list of available query items.
  destfmt -   The destination format you intend to use.

RESULT
  One of the following values is returned:
    W3D_FULLY_SUPPORTED     Completely supported by driver
    W3D_PARTIALLY_SUPPORTED Only partially supported
    W3D_NOT_SUPPORTED       Not supported

EXAMPLE

NOTES

BUGS

SEE ALSO

            W3D_Query()
            ,
            W3D_GetDrivers()
```

1.43 Warp3D/W3D_ReadStencilPixel()

```
                NAME
  W3D_ReadStencilPixel -- Read a pixel from the stencil buffer

SYNOPSIS
  success = W3D_ReadStencilPixel(context, x, y, st);
  d0                             a0       d0 d1 a1

  ULONG W3D_ReadStencilPixel(W3D_Context *, ULONG, ULONG, ULONG *);

FUNCTION
  Read the stencil buffer pixel at x,y into the variable pointed
  to by st.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - The context to use
  x,y     - Coordinates of point
  st      - Pointer to a variable to hold the read pixel

RESULT
  One of the following values:
    W3D_SUCCESS         Operation successful
    W3D_NOSTENCILBUFFER No stencil buffer present
    W3D_NOTVISIBLE      The stencil buffer can not be accessed by
                the hardware
    W3D_NOTVISIBLE      Indirect mode only. Locking failed.

EXAMPLE

NOTES
 This function is primarly intended for OpenGL implementations,
 which might need access to the stencil buffer. This function
 is slow and should normally not be called.

 Important note: In indirect mode you have to make sure, that
 the stencil buffer is up to date, no Flush is internally done
 by this function. You have to call
             W3D_Flush
             , if the stencil
 buffer is not up to date yet.

BUGS
 Indirect mode: the hardware is internally not locked for
 performance reasons, therefore the result might be wrong, if
 the corresponding buffer is swapped out.

SEE ALSO

             W3D_ReadStencilSpan
```

1.44 Warp3D/W3D_ReadStencilSpan()

```
                NAME
  W3D_ReadStencilSpan -- Read a range of stencil buffer pixels

SYNOPSIS
  success = W3D_ReadStencilSpan(context, x, y, n, st);
  d0                            a0       d0 d1 d2 a1

  ULONG W3D_ReadStencilSpan(W3D_Context *, ULONG, ULONG, ULONG,
       ULONG []);

FUNCTION
  Read a span of pixel value from the stencil buffer. The resulting
  pixels are put into the memory area pointed to by st.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - The context
  x,y     - Coordinates of span start
  n       - Number of pixels to read
  st      - pointer to the array to hold the pixel

RESULT
  One of the following values:
    W3D_SUCCESS             Operation successful
    W3D_NOSTENCILBUFFER     No stencil buffer found
    W3D_NOTVISIBLE          The stencil buffer can not be accessed by
                  the hardware
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
 If you need to read more than one consecutive pixel, use this
 function instead of calling the single pixel version repeatedly.

 This function is primarly intended for OpenGL implementations,
 which might need access to the stencil buffer. This function
 is slow and should normally not be called.

 Important note: In indirect mode you have to make sure, that
 the stencil buffer is up to date, no Flush is internally done
 by this function. You have to call
             W3D_Flush
             , if the stencil
 buffer is not up to date yet.

BUGS
 Indirect mode: the hardware is internally not locked for
 performance reasons, therefore the result might be wrong, if
 the corresponding buffer is swapped out.

SEE ALSO

             W3D_ReadStencilPixel
```

1.45 Warp3D/W3D_ReadZPixel()

```
                NAME
  W3D_ReadZPixel -- Read a pixel value from the ZBuffer

SYNOPSIS
  success = W3D_ReadZPixel(context, x, y, z);
  d0                       a0       d0 d1 a1

  ULONG W3D_ReadZPixel(W3D_Context *, ULONG, ULONG, W3D_Double *);

FUNCTION
  Read ZBuffer pixel x,y into variable pointed to by z;
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - pointer to the context
  x,y     - coordinates of pixel
  z       - pointer to a W3D_Double

RESULT
  One of the following:
    W3D_SUCCESS     Successful operation
    W3D_NOZBUFFER   No ZBuffer was allocated
    W3D_NOTVISIBLE  ZBuffer is not visible

EXAMPLE

NOTES
  This function is primarly intended for OpenGL implementations,
  which might need access to the Z buffer. This function
  is slow and should normally not be called.

  * IMPORTANT NOTE: *
  For speed reasons, this call is *NOT* compatible with indirect drawing.
  To use this call with indirect mode, you have to manually
            W3D_Flush
            ,
  and, should you use any drawing calls, you'll have to
            W3D_Flush
             again.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.

SEE ALSO

            W3D_ReadZSpan
```

1.46 Warp3D/W3D_ReadZSpan()

```
                NAME
  W3D_ReadZSpan -- read a range of ZBuffer pixels

SYNOPSIS
  success = W3D_ReadZSpan(context, x, y, n, z);
  d0                      a0       d0 d1 d2 a1

  ULONG W3D_ReadZSpan(W3D_Context *, ULONG, ULONG, ULONG, W3D_Double []);

FUNCTION
  Read a span of ZBuffer pixels into an array pointed to by the z
  parameter.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - Pointer to the context
  x,y     - Coordinates of pixels
  n       - Number of pixels to read
  z       - Array of W3D_Double to fill. Note that the array must
      be large enough (i.e. at least n)

RESULT
  One of the following values
    W3D_SUCCESS     Operation successful
    W3D_NOZBUFFER   No ZBuffer was allocated
    W3D_NOTVISIBLE  ZBuffer is not visible

EXAMPLE

NOTES
  You should use this function instead of
             W3D_ReadZPixel
              if you‘re
  going to read more pixels than just one.

  This function is primarly intended for OpenGL implementations,
  which might need access to the Z buffer. This function
  is slow and should normally not be called.

  * IMPORTANT NOTE: *
  For speed reasons, this call is *NOT* compatible with indirect drawing.
  To use this call with indirect mode, you have to manually
             W3D_Flush
             ,
  and, should you use any drawing calls, you’ll have to
             W3D_Flush
              again.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.
```

```
SEE ALSO

             W3D_ReadZPixel
```

1.47 Warp3D/W3D_ReleaseTexture()

```
                NAME
  W3D_ReleaseTexture -- Release texture from video ram

SYNOPSIS
  W3D_ReleaseTexture(context, texture);
           a0       a1

  void W3D_ReleaseTexture(W3D_Context *, W3D_Texture *);

FUNCTION
  Release a texture from video ram. This frees the memory
  allocated by that texture.

INPUTS
  context - Pointer to a W3D_Context
  texture - Pointer to the texture to be released

RESULT
  None

EXAMPLE
  extern W3D_Texture *texture;
  extern W3D_Context *context;
  W3D_ReleaseTexture(context, texture);

NOTES
  This call does nothing if W3D_AUTOTEXMANAGEMENT is set
  in the context's state.

BUGS

SEE ALSO

             W3D_UploadTexture
```

1.48 Warp3D/W3D_RequestMode()

```
NAME
  W3D_RequestMode -- Request a screen mode (V2)

SYNOPSIS
  ModeID = W3D_RequestMode(taglist);
   D0                      a0
```

```
  ULONG W3D_RequestMode(struct TagItem *);

FUNCTION
  This function presents the user with an ASL-Type screen mode
  requester. The mode requester will only include those screen modes
  that are supported by the specified combination of tag items.

INPUTS
  taglist - A taglist of W3D_SMR_#? items. The following items
      are defined:
    W3D_SMR_SIZEFILTER (BOOL)
      If set to TRUE, filter ASLSM_MinWidth, ASLSM_MinHeight,
      ASLSM_MaxWidth, ASL_MaxHeight
    W3D_SMR_DRIVER (W3D_Driver *)
      A pointer to a W3D_Driver structure that you want to use.
      If this tag is specified, the screen modes in the
      requester will all be compatible with this driver.
    W3D_SMR_DESTFMT (W3D_FMT_#? constants)
      The screen/bitmap formats you want to use. If this tag
      is active, all screenmodes will be filtered accordingly.
      You may specify a bitmask to get more than one format.
    W3D_SMR_TYPE (W3D_DRIVER_3DHW/W3D_DRIVER_CPU)
      Specifies if you want to filter the screen modes according
      to the driver type. If this is set to W3D_DRIVER_CPU,
      only the active CPU driver is used for filtering. Otherwise,
      all modes of all hardware is filtered, unless the W3D_SMR_DRIVER
      tag specifies a special driver.
    ASLSM_???
      You may give an arbitrary number of ASLSM_#? tags that will be
      passed to asl.library. Most notably, these include those tags
      the localize the requester or modify the look, including position
      and size. Most notably, the ASLSM_Min#? and ASLSM_Max#? tags
      may be used in a special meaning if the W3D_SMR_SIZEFILTER
      tag item is present and set to TRUE.

    Not all of the combinations make sense, for example, specifiying
    W3D_SMR_TYPE together with W3D_SMR_DRIVER.

RESULT
  ModeID  -   The ModeID the user selected, or INVALID_ID if the requester
        was cancelled.

EXAMPLE

NOTES

BUGS

SEE ALSO
  W3D_SelectDriver()
```

1.49 Warp3D/W3D_SetAlphaMode()

```
NAME
  W3D_SetAlpha -- Set the alpha test mode

SYNOPSIS
  success = W3D_SetAlphaMode(context, mode, refval);
  d0                         a0       d1    a1

  ULONG W3D_SetAlphaMode(W3D_Context, ULONG, W3D_Float *);

FUNCTION
  This function defines the way the alpha test is performed.
  This test compares the incoming pixel's alpha value
  with the reference value, and decides, depending on the set
  mode, if the pixel is discarded or not.

INPUTS
  context  - The context
  mode     - The alpha test mode. One of the following:
       W3D_A_NEVER       Always discard
       W3D_A_LESS        Draw, if value < refvalue
       W3D_A_GEQUAL      Draw, if value >= refvalue
       W3D_A_LEQUAL      Draw, if value <= refvalue
       W3D_A_GREATER     Draw, if value > refvalue
       W3D_A_NOTEQUAL    Draw, if value != refvalue
       W3D_A_ALWAYS      always draw
  refvalue - Pointer to the alpha reference value. Must be in
       the interval [0..1]

RESULT
  One of the following:
    W3D_SUCCESS             Success
    W3D_ILLEGALINPUT        Illegal alpha mode
    W3D_UNSUPPORTEDATEST    Alpha test unsupported
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Alpha testing is probably not supported on older 3D hardware.

BUGS

SEE ALSO
```

1.50 Warp3D/W3D_SetBlendMode()

```
                NAME
  W3D_SetBlendMode -- Set the blending mode

SYNOPSIS
  success = W3D_SetBlendMode(context, srcfunc, dstfunc);
  d0                         a0       d0       d1

  ULONG W3D_SetBlendMode(W3D_Context *, ULONG, ULONG);
```

```
FUNCTION
  Sets the blending mode. Blending has to be enabled using

             W3D_SetState
             . For more information about the blending modes, see
  the OpenGL specs.

INPUTS
  context - pointer to the W3D_Context
  srcfunc - The mode for the source pixel. Values are:
        W3D_ZERO
        W3D_ONE
        W3D_DST_COLOR
        W3D_ONE_MINUS_DST_COLOR
        W3D_SRC_ALPHA
        W3D_ONE_MINUS_SRC_ALPHA
        W3D_DST_ALPHA
        W3D_ONE_MINUS_DST_ALPHA
        W3D_SRC_ALPHA_SATURATE
        W3D_CONSTANT_COLOR
        W3D_ONE_MINUS_CONSTANT_COLOR
        W3D_CONSTANT_ALPHA
        W3D_ONE_MINUS_CONSTANT_ALPHA
  dstfunc - Mode for the destination:
        W3D_ZERO
        W3D_ONE
        W3D_SRC_COLOR
        W3D_ONE_MINUS_SRC_COLOR
        W3D_SRC_ALPHA
        W3D_ONE_MINUS_SRC_ALPHA
        W3D_DST_ALPHA
        W3D_ONE_MINUS_DST_ALPHA
        W3D_CONSTANT_COLOR
        W3D_ONE_MINUS_CONSTANT_COLOR
        W3D_CONSTANT_ALPHA
        W3D_ONE_MINUS_CONSTANT_ALPHA

RESULT
  One of the following:
    W3D_SUCCESS             Success
    W3D_ILLEGALINPUT        Illegal alpha blend mode
    W3D_UNSUPPORTEDBLEND    Mode is not supported by current driver
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_SetState
             ,
             W3D_GetState
```

1.51 Warp3D/W3D_SetChromaTestBounds()

```
   NAME
     W3D_SetChromaTestBounds -- Set range for color keying

   SYNOPSIS
     res = W3D_SetChromaTestBounds(context, texture, lower, upper, mode)
     d0                            a0       a1       d0     d1     d2

     ULONG W3D_SetChromaTestBounds(W3D_Context *, W3D_Texture *, ULONG, ULONG,  ↩
        ULON
G);

   FUNCTION
     Sets the bounds for chroma testing. All colors inside the range defined by
     bounds are treated normally, while pixels outside the range are not drawn.

   INPUTS
     context - pointer to a context to use
     texture - pointer to a texture
     lower   - lower bound.
     upper   - upper bound
     mode    - chroma test mode
         The following values are possible:
         W3D_CHROMATEST_NONE         disable chroma testing
         W3D_CHROMATEST_INCLUSIVE    texels within the specified range pass
                        the test (i.e. get drawn)
         W3D_CHROMATEST_EXCLUSIVE    only texels outside the specified range
                        are drawn, others are rejected.

   RESULT
     One of the following:
       W3D_SUCCESS             Success
       W3D_UNSUPPORTED        Chroma keying is not supported on this hardware

   EXAMPLE

   NOTES

   BUGS

   SEE ALSO
```

1.52 Warp3D/W3D_SetColorMask()

```
                  NAME
     W3D_SetColorMask -- Set mask for drawing

   SYNOPSIS
     success = W3D_SetColorMask(context, red, green, blue, alpha);
     d0                         a0       d0   d1     d2    d3

     ULONG W3D_SetColorMask(W3D_Context *, W3D_Bool, W3D_Bool, W3D_Bool,
       W3D_Bool);
```

```
FUNCTION
  This function defines the mask for all drawing operations in
  direct color mode (15/16/24/32 bit modes).

INPUTS
  context     - the context
  red
  green
  blue
  alpha       - If set to FALSE, the component should be masked out.

RESULT
  W3D_SUCCESS             Success
  W3D_MASKNOTSUPPORTED    Masking is not supported by the current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_SetPenMask
```

1.53 Warp3D/W3D_SetCurrentColor()

```
NAME
  W3D_SetCurrentColor -- Set color for single-color operations

SYNOPSIS
  ret = W3D_SetCurrentColor(context, color);
              a0       a1

  ULONG W3D_SetCurrentColor(W3D_Context *, W3D_Color *);

FUNCTION
  Defines the color to use for operations where one single color
  is used, i.e. flat-shaded opbjects. This color is only used for
  RGBA destinations.

INPUTS
  context - Context pointer
  color   - Pointer to a color to use

RESULT
  W3D_QUEUFAIL    Queueing failed in indirect mode
  W3D_NOTVISIBLE  Locking failed in indirect mode

EXAMPLE

NOTES
```

```
BUGS

SEE ALSO
```

1.54 Warp3D/W3D_SetCurrentPen()

```
NAME
  W3D_SetCurrentPen -- Set pen for single-color operations

SYNOPSIS
  W3D_SetCurrentPen(context, pen);
          a0       d1

  void W3D_SetCurrentPen(W3D_Context *, ULONG);

FUNCTION
  Define the pen to use for single-color operations, such as flat-shaded
  objects. The pen setting is olny used for chunky destinations.

INPUTS
  context - a context pointer
  pen     - the pen number to use

RESULT
  W3D_QUEUFAIL    Queueing failed in indirect mode
  W3D_NOTVISIBLE  Locking failed in indirect mode

EXAMPLE

NOTES

BUGS

SEE ALSO
```

1.55 Warp3D/W3D_SetDrawRegion()

```
NAME
  W3D_SetDrawRegion -- Set the clipping rectangle

SYNOPSIS
  success = W3D_SetDrawRegion(context, bm, yoffset, scissor);
  d0                          a0       a1  d1       a2

  ULONG W3D_SetDrawRegion(W3D_Context *, struct BitMap *, ULONG,
      W3D_Scissor *);

FUNCTION
  This function defines/changes the current drawing region.
  It's used for multibuffering and clipping.
```

```
INPUTS
  context - The context
  bm      - The bitmap to draw to. If NULL, the old bitmap is used
  yoffset - The vertical offset for the top-left edge. Used for
      multibuffering.
  scissor - If not NULL, defines the scissoring region. All values
      are taken to be relative to (0, yoffset) in the bitmap.


RESULT
  One of the following:
    W3D_SUCCESS         Success.
    W3D_ILLEGALBITMAP   Illegal bitmap
    W3D_UNSUPPORTEDFMT  Unsupported format
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Due to constraints on bitmap placement in some drivers, bitmap data
  must be aligned to 8 byte boundaries

BUGS

SEE ALSO
```

1.56 Warp3D/W3D_SetDrawRegionWBM()

```
               NAME
  W3D_SetDrawRegionWBM -- Set the clipping rectangle for a W3D_Bitmap

SYNOPSIS
  success =
            W3D_SetDrawRegion
            (context, bm, scissor);
  d0                          a0       a1  a2

  ULONG
            W3D_SetDrawRegion
            (W3D_Context *, W3D_Bitmap *, W3D_Scissor *);

FUNCTION
  This function defines/changes the current drawing region.
  It's used for multibuffering and clipping.
  The only difference to
            W3D_SetDrawRegion
             is the bitmap used.

INPUTS
  context - The context
  bm      - The bitmap to draw to. If NULL, the old bitmap is used
  scissor - If not NULL, defines the scissoring region. All values
      are taken to be relative to (0, yoffset) in the bitmap.

RESULT
```

```
    One of the following:
      W3D_SUCCESS         Success.
      W3D_ILLEGALBITMAP   Illegal bitmap
      W3D_UNSUPPORTEDFMT  Unsupported format

  EXAMPLE

  NOTES

  BUGS

  SEE ALSO

              W3D_SetDrawRegion
```

1.57 Warp3D/W3D_SetFilter()

```
                  NAME
    W3D_SetFilter -- Set the filter method

  SYNOPSIS
    res = W3D_SetFilter(context, texture, MinFilter, MagFilter);
    d0                  a0       a1       d0         d1

    ULONG W3D_SetFilter(W3D_Context *, W3D_Texture *, ULONG,
        ULONG);

  FUNCTION
    Set the texture's filter mode. The filter mode used is
    texture dependant, so it is possible to set different
    filter modes for different texture.

  INPUTS
    context     - Pointer to a W3D_Context
    texture     - Pointer to the texture to be modified
    MinFilter   - Minification filter. May be one of the following:
          W3D_NEAREST               no mipmapping, no filtering
          W3D_LINEAR                no mipmapping, bilinear filtering
          W3D_NEAREST_MIP_NEAREST   mippmapping, no filtering
          W3D_LINEAR_MIP_NEAREST    mipmapping, bilinear filtering
          W3D_NEAREST_MIP_LINEAR    mipmapping filtered, no filtering on t
exture
          W3D_LINEAR_MIP_LINEAR     mippmapping with trilinear filtering
    MagFilter   - Magnification filter. One of these:
          W3D_NEAREST               no filtering
          W3D_LINEAR                Bilinear filtering

  RESULT
    A value indicating success of failure. May be one of the following:
      W3D_SUCCESS             Success
      W3D_ILLEGALINPUT        Illegal values for Min/MagFilter
      W3D_UNSUPPORTEDFILTER   Desired filter not supported by driver
      W3D_WARNING             Success, but the filter mode was adjusted,
                  because *_MIP_* was given for a texture
```

```
                without mipmaps
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Some hardware may ignore the MagFilter. In this case, the MinFilter
  is used even if the texture is enlarged.

BUGS

SEE ALSO

             W3D_Query
             ,
             W3D_GetTexFmtInfo
```

1.58 Warp3D/W3D_SetFogParams()

```
NAME
  W3D_SetFogParams -- Set fog parameters

SYNOPSIS
  success = W3D_SetFogParams(context, fogparams, fogmode);
  d0                          a0       a1         d1

  ULONG W3D_SetFogParams(W3D_Context *, W3D_Fog *, ULONG);

FUNCTION
  This function defines fogging parameters and modes.

INPUTS
  context     - The context to be modified
  fogparams   - Pointer to a W3D_Fog.
  fogmode     - The type of fog.
          W3D_FOG_LINEAR  Linear fog
          W3D_FOG_EXP     Exponential fog
          W3D_FOG_EXP_2   Square exponential fogging

RESULT
  One of the following:
    W3D_SUCCESS         Success
    W3D_ILLEGALINPUT    Illegal input
    W3D_UNSUPPORTEDFOG  Fog mode is not supported by current driver
    W3D_NOTVISIBLE      Indirect mode only. Locking failed.

EXAMPLE

NOTES

BUGS

SEE ALSO
```

1.59 Warp3D/W3D_SetLogicOp()

```
NAME
  W3D_SetLogicOp -- Define logical operation

SYNOPSIS
  success = W3D_SetLogicOp(context, operation);
  d0                       a0       d1

  ULONG W3D_SetLogicOp(W3D_Context *, ULONG);

FUNCTION
  Set the logical operation. For further information, see the OpenGL
  specs.

INPUTS
  context     - Same as ever
  operation   - The logical operation desired. Possible values are:
          W3D_LO_CLEAR            dest = 0
          W3D_LO_AND              dest = source & dest
          W3D_LO_AND_REVERSE      dest = source & !dest
          W3D_LO_COPY             dest = source
          W3D_LO_AND_INVERTED     dest = !source & dest
          W3D_LO_NOOP             dest = dest
          W3D_LO_XOR              dest = source ^ dest
          W3D_LO_OR               dest = source | dest
          W3D_LO_NOR              dest = !(source | dest)
          W3D_LO_EQUIV            dest = !(source ^ dest)
          W3D_LO_INVERT           dest = !dest
          W3D_LO_OR_REVERSE       dest = source | !dest
          W3D_LO_COPY_INVERTED    dest = !source
          W3D_LO_OR_INVERTED      dest = !source | dest
          W3D_LO_NAND             dest = !(source & dest)
          W3D_LO_SET              dest = 1

RESULT
  W3D_SUCCESS             Success
  W3D_ILLEGALINPUT        Wrong operation
  W3D_UNSUPPORTEDLOGICOP  Unsupported by current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES

BUGS

SEE ALSO
```

1.60 Warp3D/W3D_SetPenMask()

```
                NAME
  W3D_SetPenMask -- set a pen mask for drawing operations
```

```
SYNOPSIS
  ret = W3D_SetPenMask(context, indexmask)
  d0                   a0       d1

  ULONG W3D_SetPenMask(W3D_Context *, ULONG);

FUNCTION
  This function defines the mask for all drawing operations in
  chunky modes (8 bit modes).

INPUTS
  context     - The context to use
  indexmask   - A bitmask which is applied to chunky pixels

RESULT
  W3D_SUCCESS             Success
  W3D_MASKNOTSUPPORTED    Masking is not supported by the current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES

BUGS

SEE ALSO

             W3D_SetColorMask
```

1.61 Warp3D/W3D_SetScissor()

```
                NAME
  W3D_SetScissor -- (Re-) Set the clipping rectangle

SYNOPSIS
  W3D_SetScissor(context,scissor);
          a0      a1

  void W3D_SetScissor(W3D_Context* context, W3D_Scissor* scissor);

FUNCTION
  This function sets or resets the clipping rectangle while retaining
  the current drawing region.

INPUTS
  context     -   The context structure
  scissor     -   A new scissor or NULL for full-screen/no clipping

RESULT

EXAMPLE

NOTES
```

```
   BUGS

   SEE ALSO

                W3D_SetDrawRegion()
```

1.62 Warp3D/W3D_SetState()

```
                    NAME
     W3D_SetState -- Enable or disable hardware and context states

   SYNOPSIS
     success = W3D_SetState(context, state, newstate);
     d0                     a0       d0     d1

     ULONG W3D_SetState(W3D_Context *, ULONG, ULONG);

   FUNCTION
     This function is used to enable or disable hardware
     effects or context states. Success or failure depends
     on the hardware's ability to use the effect. Some
     hardware may not even be able to switch off some effects.

   INPUTS
     context     - pointer to a W3D_Context
     state       - state to be changed. Current states are listed here.
           For a more detailed description, read the doc files.
             W3D_AUTOTEXMANAGEMENT  automatic texture management
             W3D_SYNCHRON           wait, until HW is idle
             W3D_INDIRECT           buffer drawings until
                W3D_Flush()
                'ed
             W3D_GLOBALTEXENV       global texture modes
             W3D_DOUBLEHEIGHT       screen has double height
             W3D_FAST               Drawing functions may modify passed str
uctures
             W3D_TEXMAPPING         texmapping state
             W3D_PERSPECTIVE        perspective correction state
             W3D_GOURAUD            gouraud/flat shading
             W3D_ZBUFFER            Z-Buffer state
             W3D_ZBUFFERUPDATE      Z-Buffer update state
             W3D_BLENDING           Alpha blending state
             W3D_FOGGING            Fogging state
             W3D_ANTI_POINT         Point antialiasing
             W3D_ANTI_LINE          Line antialiasing
             W3D_ANTI_POLYGON       Polygon antialiasing
             W3D_ANTI_FULLSCREEN    Fullscreen antialiasing
             W3D_DITHERING          dithering state
             W3D_LOGICOP            logical operations state
             W3D_STENCILBUFFER      stencil buffer state
             W3D_ALPHATEST          alpha test operation
             W3D_SPECULAR           Specular highlightung state
             W3D_TEXMAPPING3D       3D texturemapping state
             W3D_SCISSOR            Scissor test
```

```
                W3D_CHROMATEST            Chroma testing (i.e. color k
eying)
     newstate    - indicates what should be done to the state bit:
             W3D_ENABLE             try to switch this feature on
             W3D_DISABLE            try to switch it off

   RESULT
     One of two constants:
       W3D_SUCCESS            the operation was successful
       W3D_UNSUPPORTEDSTATE   the operation can not be done

   EXAMPLE
     if (W3D_UNSUPPORTEDSTATE == W3D_SetState(context, W3D_ANTI_FULLSCREEN,
                    W3D_ENABLE)) {
       printf("This hardware does not support fullscreen antialiasing\n");
     } else {
       printf("Fullscreen antialiasing enabled\n");
     }

   NOTES
     It's not required to check the return value, however, do not assume anything.
     The current hardware may not have any restrictions on using
     i.e. Z buffering, but future hardware may.

   BUGS

   SEE ALSO

              W3D_GetState
              ,
              W3D_Query
```

1.63 Warp3D/W3D_SetStencilFunc()

```
                 NAME
     W3D_SetStencilFunc -- Set stencil function

   SYNOPSIS
     success = W3D_SetStencilFunc(context, func, refvalue, mask);
     d0                           a0       d0    d1        d2

     ULONG W3D_SetStencilMode(W3D_Context *, ULONG, ULONG, ULONG);

   FUNCTION
     Set the stencil test function, as used by the OpenGL render pipeline.
     For more information, refer to the OpenGL specs.

   INPUTS
     context         - W3D context structure
     func            - stencil test function. Possible value are:
               W3D_ST_NEVER         don't draw pixel
               W3D_ST_ALWAYS        draw always
               W3D_ST_LESS          draw, if refvalue < ST
               W3D_ST_LEQUAL        draw, if refvalue <= ST
```

```
            W3D_ST_EQUAL         draw, if refvalue == ST
            W3D_ST_GEQUAL        draw, if refvalue >= ST
            W3D_ST_GREATER       draw, if refvalue > ST
            W3D_ST_NOTEQUAL      draw, if refvalue != ST
  refvalue        - reference value (0-255) used for the stencil test
  mask            - mask value applied to 'refvalue' and to the stencil buffer
          content

RESULT
  W3D_SUCCESS             Success
  W3D_ILLEGALINPUT        Illegal input
  W3D_UNSUPPORTEDSTTEST   Not supported by current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Stencil buffering is only supported by newer hardware
  Note that the stencil test has to be enabled using

             W3D_SetState
             .

BUGS

SEE ALSO
```

1.64 Warp3D/W3D_SetStencilOp()

```
                NAME
  W3D_SetStencilOp -- Set stencil operation

SYNOPSIS
  success = W3D_SetStencilOp(context, sfail, dpfail, dppass);
  d0                         a0       d0     d1      d2

  ULONG W3D_SetStencilOp(W3D_Context *, ULONG, ULONG, ULONG);

FUNCTION
  Set the stencil test operation, as used by the OpenGL render
  pipeline. For more information, refer to the OpenGL specs.

INPUTS
  context         - context pointer
  dpfail          - action, if depth test fails
  dppass          - action, if depth test succeeds. Possible values are
          (for all three mentioned cases):
            W3D_ST_KEEP          keep stencil buffer value
            W3D_ST_ZERO          clear stencil buffer value
            W3D_ST_REPLACE       replace by reference value
            W3D_ST_INCR          increment
            W3D_ST_DECR          decrement
            W3D_ST_INVERT        invert bitwise
```

```
RESULT
  W3D_SUCCESS             Success
  W3D_ILLEGALINPUT        Illegal input
  W3D_UNSUPPORTEDSTTEST   Not supported by current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Stencil buffering is only supported on newer hardware.
  Note that the stencil test has to be enabled using

             W3D_SetState
             .

BUGS

SEE ALSO
```

1.65 Warp3D/W3D_SetTexEnv()

```
                NAME
  W3D_SetTexEnv -- Set texture environment parameters

SYNOPSIS
  success = W3D_SetTexEnv(context, texture, envparam, envcolor);
  d0                      a0       a1       d1        a2

  ULONG W3D_SetTexEnv(W3D_Context *, W3D_Texture *, ULONG,
      W3D_Color *);

FUNCTION
  This function is used to set the texture environment parameters.
  These parameters define how a texture is applied to a drawn
  primitive. This also involves lit-texturing, and unlit-texturing.

INPUTS
  context     - a pointer to a W3D_Context (surprise !:)
  texture     - a pointer to the texture object to be modified
  envparam    - the environment parameter. One of the following:
        W3D_REPLACE       Unlit texturing
        W3D_DECAL         Lit texturing using the alpha component
                  as blending value
        W3D_MODULATE      Lit texturing by modulation of source
                  and destination. Modulation means
                  source and destination are multiplied.
        W3D_BLEND         Blending with the color in envcolor.
  envcolor    - Only specified when envparam == W3D_BLEND. The
        given color value is used for blending with the texture.
        Must be NULL for all other envparams.

RESULT
  A value indicating success or failure. Current values are:
    W3D_SUCCESS             (guess :)
```

```
    W3D_ILLEGALINPUT        Unknown envparam given
    W3D_UNSUPPORTEDTEXENV   Not supported by the current driver
    W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
 The texture environment is texture-specific by default. By enabling
 the W3D_GLOBALTEXENV state using
             W3D_SetState()
              the texture environment
 can be made global for all textures (this is the case in OpenGL,
 for example).

BUGS

SEE ALSO

             W3D_GetTexFmtInfo
```

1.66 Warp3D/W3D_SetWrapMode()

```
                NAME
  W3D_SetWrapMode -- Set the texture's wrapping mode

SYNOPSIS
  success = W3D_SetWrapMode(context, texture, mode_s, mode_t, border);
  d0                        a0        a1      d0      d1      a2

  ULONG W3D_SetWrapMode(W3D_Context *, W3D_Texture *, ULONG,
      ULONG, W3D_Color *);

FUNCTION
  Sets the texture's wrapping mode.

INPUTS
  context     - A W3D_Context pointer
  texture     - The texture to be modified
  mode_s      - The wrapping in s direction (vertical). Can be one
        of the following constants:
        W3D_REPEAT    Texture is repeated
        W3D_CLAMP     Texture is clamped, the border is filled
                with the color given in border.
  mode_t      - Wrapping in t direction (horizontal). Same as above.
  border      - A pointer to a W3D_Color used for the border (when clamping).

RESULT
  A value indicating success or failure. One of the following:
    W3D_SUCCESS             - Success
    W3D_ILLEGALINPUT        - Illegal wrap mode
    W3D_UNSUPPORTEDWRAPMODE - The desired wrap mode is not supported
                by the current driver

EXAMPLE
```

```
NOTES
 The Virge does not allow asymmetric wrapping, therefore you should
 use the query facility, if asymmetric wrapping is possible.

 You should usually use W3D_REPEAT, since W3D_CLAMP is currently
 not possible with the Virge.

BUGS

SEE ALSO

             W3D_Query
             ,
             W3D_GetTexFmtInfo
```

1.67 Warp3D/W3D_SetWriteMask()

```
                NAME
  W3D_SetWriteMask -- write protext bits in the stencil buffer

SYNOPSIS
  success = W3D_SetWriteMask(context, mask);
  d0                         a0       d1

  ULONG W3D_SetWriteMask(W3D_Context *, ULONG);

FUNCTION
  Defines which bits of the stencil buffer are write protected

INPUTS
  context - context pointer
  mask    - a bitmask, indicationg which bits of the
      stencil buffer should be write-protected.
      Setting a bit to 1 allows write access,
      while a 0 bit protects it from writing

RESULT
  W3D_SUCCESS             success
  W3D_UNSOPPORTEDTEST     Not supported by current driver
  W3D_NOTVISIBLE          Indirect mode only. Locking failed.

EXAMPLE

NOTES
  Stencil buffering is only supported on newer hardware.
  Note that the stencil test has to be enabled using

             W3D_SetState
             .

BUGS

SEE ALSO
```

1.68 Warp3D/W3D_SetZCompareMode()

```
                NAME
  W3D_SetZCompareMode -- Set the ZBuffer compare mode

SYNOPSIS
  success = W3D_SetZCompareMode(context, mode);
  d0                            a0       d1

  ULONG W3D_SetZCompareMode(W3d_Context *, ULONG);

FUNCTION
  Set the compare mode used by ZBuffering. This mode
  determines what will be drawn depending on the z coordinate
  of the primitive to be drawn, and the value currently
  in the ZBuffer. For more information on ZBuffering, see the
  OpenGL specs, or get a textbook about Computer Graphics.

INPUTS
  context - A context pointer
  mode    - The ZBuffer compare mode. One of the following values:
        W3D_Z_NEVER             Never pass, discard pixel
        W3D_Z_LESS              Draw if z < zbuffer
        W3D_Z_GEQUAL            Draw if z >= zbuffer
        W3D_Z_LEQUAL            Draw if z <= zbuffer
        W3D_Z_GREATER           Draw if z > zbuffer
        W3D_Z_NOTEQUAL          Draw if z != zbuffer
        W3D_Z_EQUAL             Draw if Z == zbuffer
        W3D_Z_ALWAYS            Always draw

RESULT
  One of the following values:
    W3D_SUCCESS         Operation successful
    W3D_ILLEGLAINPUT    Illegal compare mode
    W3D_UNSUPPORTEDZCMP Comparemode unsupported by current driver
    W3D_NOTVISIBLE      Indirect mode only. Locking failed.

EXAMPLE

NOTES
  W3D_Z_LESS is the "normal" behavior (i.e. depth cueing), while
  W3D_Z_NOTEQUAL can be used as a poor man's stencil buffering.

  When mixing software and hardware rendering (for example in OpenGL
  implementations, then you should be aware, that using some of
  the Z compare modes (i.e. W3D_Z_EQUAL, W3D_Z_NOTEQUAL) might not
  work correctly, since the results of the software engine might
  not be exactly the same as the results of the hardware engine.


BUGS

SEE ALSO
```

```
              W3D_ClearZBuffer
```

1.69 Warp3D/W3D_TestMode()

```
                NAME
  W3D_TestMode -- Test Mode and return driver (V2)

SYNOPSIS
  driver = W3D_TestMode(modeid);
   D0                   D0

  W3D_Driver *W3D_TestMode(ULONG);

FUNCTION
  Given a standard ModeID, this function tests if there is a
  driver available for this DisplayID. A hardware driver is
  preferred, although it will return a CPU driver (if found)
  in case none of the installed hardware drivers support this
  screenmode.

INPUTS
  modeid  -   A standard AmigaOS DisplayID

RESULT
  driver  -   A pointer to a suitable driver or NULL if
        no matching or CPU driver found.

EXAMPLE

NOTES
  This function will also check if the CPU driver actually supports
  this format, so be prepared to check for a NULL return value.

BUGS

SEE ALSO

              W3D_GetDrivers
```

1.70 Warp3D/W3D_UnLockHardware()

```
                NAME
  W3D_UnLockHardware -- Release the exclusive hardware lock

SYNOPSIS
  W3D_UnLockHardware(context);
           a0

  void W3D_UnLockHardware(W3D_Context *);
```

```
FUNCTION
  This function releases a hardware lock previously acquired
  with
             W3D_LockHardware
             .

INPUTS
  context - a pointer to a W3D_Context

RESULT
  None

EXAMPLE
  if (W3D_SUCCESS ==
             W3D_LockHardware
             (context) {
    ...
    Render some stuff
    ...
    W3D_UnLockHardware(context);
  } else {
    printf("Can't lock hardware\n");
  }

NOTES

BUGS

SEE ALSO

             W3D_LockHardware
             ,
             W3D_GetState
```

1.71 Warp3D/W3D_UpdateTexImage()

```
                NAME
  W3D_UpdateTexImage -- Change the image of a texture or mipmap

SYNOPSIS
  success = W3D_UpdateTexImage(context, texture, teximage, level, palette);
  d0                           a0       a1       a2        d1     a3

  ULONG W3D_UpdateTexImage(W3D_Context *, W3D_Texture *, void *,
      ULONG, ULONG *);

FUNCTION
  Change the image mipmap data to the given texture. The new source
  image must have dimensions and format equal to the old one. Also,
  mipmap mode must be the same (meaning that if the old texture had
  mipmaps, so must the new).
  The resident state is unaffected. If the texture is in video ram,
  the copy there will be replaced by the new image as soon as the
```

```
  texture is used again for rendering.

INPUTS
  context     - a pointer to the current context
  texture     - a pointer to the texture to be modified
  teximage    - a pointer to the new image data
  level       - the texture level to be changed. 0 is the source image,
        while levels != 0 are the mipmaps.
  palette     - a pointer to a palette, if needed. May be NULL, even if
        the texture is chunky, in which case the old palette
        will remain valid. See the note to the W3D_ATO_PALETTE
        tag in W3D_AllocTexObject for some constraints on using
        chunky textures on 8bit screens

RESULT
  One of the following:
    W3D_SUCCESS     Success
    W3D_NOMEMORY    No memory left
    W3D_NOMIPMAPS   Mipmaps are not supported by this texture object
    W3D_NOTVISIBLE  (Indirect context only) Flushing failed due to failed
            hardware locking

EXAMPLE

NOTES
 Update operations are expensive, when done very often, because of
 the bus bandwidth limitation. Be especially careful when using
 texture animations. On hardware with a lot of VRAM, it might be
 better to treat all frames of such an animation as separate
 textures, so that all (or most of them) might be in VRAM.

BUGS

SEE ALSO

             W3D_AllocTexObj
```

1.72 Warp3D/W3D_UpdateTexSubImage()

```
                NAME
  W3D_UpdateTexSubImage -- Change part of a texture

SYNOPSIS
  success = W3D_UpdateTexSubImage(context, texture, teximage, level,
  d0                              a0       a1       a2        d1
         palette, scissor, srcbpr);
         a3       a4       d0

  ULONG
            W3D_UpdateTexImage
            (W3D_Context *, W3D_Texture *, void *,
      ULONG, ULONG *, W3D_Scissor*, ULONG);

FUNCTION
```

```
  Update only part of a texture, as defined by the scissor region.
  The image data is assumed to be as large as the scissor region.
  If it's larger, the srcbpr parameter can be used to define the number
  of bytes per source row. If teximage is non-zero, the contents is copied
  into the texture. It can also be set to NULL. In this case, you can alter
  the texture image yourself in the following way: The pointer supplied with
  W3D_AllocTexObj/W3D_UpdateTexImage points to your supplied image data. You
  are allowed to change this data, BUT you MUST call W3D_UpdateTexSubImage
  after changing BEFORE doing anything else. This call must not be used inside
  a W3D_LockHardware/W3D_UnLockHardware pair. The scissor is then
  considered a "damage region", and the area defined by it will be updated.

  This function also recreates mipmaps, also only restricted to the scissor
  region.

INPUTS
  context     - a pointer to the current context
  texture     - a pointer to the texture to be modified
  teximage    - a pointer to the new image data. Note that this pointer
        is only "temporary", it may be reused immediatly. This is
        different from the
             W3D_UpdateTexImage
              call.
  level       - the texture level to be changed. 0 is the source image,
        while levels != 0 are the mipmaps.
  palette     - a pointer to a palette, if needed. May be NULL, even if
        the texture is chunky, in which case the old palette
        will remain valid. See the note to the W3D_ATO_PALETTE
        tag in W3D_AllocTexObject for some constraints on using
        chunky textures on 8bit screens
  scissor     - The given image data will be transfered into this region.
  srcbpr      - Bytes per row in source image. May be set to zero to indicate
        that image data ans scissor size match.

RESULT
  One of the following:
    W3D_SUCCESS     Success
    W3D_NOMEMORY    No memory left
    W3D_NOMIPMAPS   Mipmaps are not supported by this texture object,
            or no mipmaps have been created yet.
    W3D_NOTVISIBLE  (Indirect context only) Flushing failed due to failed
            hardware locking

EXAMPLE

NOTES
 Update operations are expensive, when done very often, because of
 the bus bandwidth limitation. Be especially careful when using
 texture animations. On hardware with a lot of VRAM, it might be
 better to treat all frames of such an animation as separate
 textures, so that all (or most of them) might be in VRAM.

BUGS

SEE ALSO

              W3D_AllocTexObj
```

```
              ,
              W3D_UpdateTexImage
```

1.73 Warp3D/W3D_UploadTexture()

```
                 NAME
  W3D_UploadTexture -- Transfer a texture to video ram

SYNOPSIS
  success = W3D_UploadTexture(context, texture);
  d0                          a0       a1

  ULONG W3D_UploadTexture(W3D_Context *, W3D_Texture *);

FUNCTION
  `Upload` a texture to video ram. Video memory is allocated and
  the texture image is copied there. The source texture stays in
  main memory.

INPUTS
  context - a W3D_Context
  texture - the W3D_Texture to be transfered

RESULT
  A value indication success or failure. One of the following:
    W3D_SUCCESS     It worked.
    W3D_NOGFXMEM    No video ram remaining.

EXAMPLE

NOTES
  This function does nothing when W3D_AUTOTEXMANAGEMENT is set
  in the current context`s state. Note also that transferring
  textures to video ram means transfer over the hardware`s bus
  system. Although newer cards like the CVPPC will have a PCI
  or similar bus, those bus system are still considered
  `bottlenecks`, and are usually much slower than main memory
  transfers. It is advised that you use automatic texture management,
  as this uses a LRU caching scheme. This was also used in
  ADescent, and gave about 99.7 % hit ratio.

BUGS

SEE ALSO

              W3D_ReleaseTexture
              , W3D_FlushTexture.
```

1.74 Warp3D/W3D_WaitIdle()

```
                NAME
  W3D_WaitIdle -- Wait for the hardware to become idle

SYNOPSIS
  W3D_WaitIdle(context);
       a0

  void W3D_WaitIdle(W3D_Context *);

FUNCTION
  This function waits for the hardware to finish it's current
  operation. It blocks your program until then.

INPUTS
  context - a pointer to W3D_Context

RESULT
  None

EXAMPLE
  W3D_DrawSomething(context);
  W3D_WaitIdle(context);
  printf("Hardware is free again\n");

NOTES
  You should use this function instead of
             W3D_CheckIdle
              if you
  just want to wait for the hardware. This function may use
  signals and/or interrupts for waiting, letting the CPU take care
  of other tasks while waiting

  Usually you won't need to call this function, since W3D takes care,
  that any drawing operation is only done, if the hardware is
  ready to get a new job.

BUGS

SEE ALSO

              W3D_CheckIdle
```

1.75 Warp3D/W3D_WriteStencilPixel()

```
                NAME
  W3D_WriteStencilPixel -- Write a pixel into the stencil buffer

SYNOPSIS
  res = W3D_WriteStencilBuffer(context, x, y, st);
  d0                           a0       d0 d1 d2

  ULONG W3D_WriteStencilBuffer(W3D_Context *, ULONG, ULONG, ULONG);
```

```
FUNCTION
  This function writes the pixel st into the stencil buffer of context,
  at position x,y.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - a context pointer
  x,y     - position to write to
  st      - the pixel value

RESULT
  A constant indicating success or failure. One of the following:
    W3D_SUCCESS             Success
    W3D_NOSTENCILBUFFER     Stencil buffering not supported by
                current driver
    W3D_NOTVISIBLE          The stencil buffer can not be accessed by
                  the hardware

EXAMPLE

NOTES
  Stencil buffering is not supported on older hardware.

  This function is primarly intended for OpenGL implementations,
  which might need access to the stencil buffer. This function
  is slow and should normally not be called.

  Important note: In indirect mode you have to make sure, that
  the stencil buffer is up to date, no Flush is internally done
  by this function. You have to call
            W3D_Flush
            , if the stencil
  buffer is not up to date yet.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.

SEE ALSO

             W3D_AllocStencilBuffer
```

1.76 Warp3D/W3D_WriteStencilSpan()

```
                NAME
  W3D_WriteStencilSpan -- Write a span of stencil pixels

SYNOPSIS
  success = W3D_WriteStencilSpan(context, x, y, n, st, mask);
  d0                             a0       d0 d1 d2 a1  a2

  ULONG W3D_WriteStencilSpan(W3D_Context *, ULONG, ULONG, ULONG,
```

```
          ULONG [], UBYTE []);

FUNCTION
  Write a span of n stencil pixels into the stencil buffer, starting
  at x,y. Pixels are taken from st. The mask array is used to skip pixels:
  If a byte is set to 0, the corresponding pixel is not written.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - a context pointer
  x,y     - starting coordinates
  n       - number of pixels
  st      - array of stencil pixels
  mask    - mask array. May be NULL

RESULT
  A constant indicating success or failure. One of the following:
    W3D_SUCCESS             Success
    W3D_NOSTENCILBUFFER     Stencil buffering not supported by
                current driver
    W3D_NOTVISIBLE          The stencil buffer can not be accessed by
                  the hardware

EXAMPLE

NOTES
  Stencil buffering is not supported on older hardware.

  This function is primarly intended for OpenGL implementations,
  which might need access to the stencil buffer. This function
  is slow and should normally not be called.

  Important note: In indirect mode you have to make sure, that
  the stencil buffer is up to date, no Flush is internally done
  by this function. You have to call
            W3D_Flush
            , if the stencil
  buffer is not up to date yet.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.

SEE ALSO
```

1.77 Warp3D/W3D_WriteZPixel()

```
                NAME
  W3D_WriteZPixel -- Write a pixel into the ZBuffer

SYNOPSIS
  success = W3D_WriteZPixel(context, x, y, z);
```

```
  d0                        a0       d0 d1 a1

  ULONG W3D_WriteZBuffer(W3D_Context *, ULONG, ULONG, W3D_Double *);

FUNCTION
  Write  ZBuffer pixel z into context's ZBuffer, at x,y.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - The context
  x,y     - Coordinates of the pixel
  z       - Pointer to a W3D_Double that's put into the zbuffer

RESULT

EXAMPLE

NOTES
  This function is primarly intended for OpenGL implementations,
  which might need access to the Z buffer. This function
  is slow and should normally not be called.

  * IMPORTANT NOTE: *
  For speed reasons, this call is *NOT* compatible with indirect drawing.
  To use this call with indirect mode, you have to manually
            W3D_Flush
            ,
  and, should you use any drawing calls, you'll have to
            W3D_Flush
             again.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.

SEE ALSO
```

1.78 Warp3D/W3D_WriteZSpan()

```
                NAME
  W3D_WriteZSpan -- Write a span of z pixels

SYNOPSIS
  W3D_WriteZSpan(context, x, y, n, z, mask);
         a0       d0 d1 d2 a1 a2

  W3D_WriteZSpan(W3D_Context *, ULONG, ULONG, ULONG,
        W3D_Double [], UBYTE []);

FUNCTION
  Write a span of pixels pointed to by z into the zbuffer.
  Writing begins at x,y, n pixels will be drawn. mask points
```

```
  to an equally sized array of UBYTES. A 0 in the array indicates
  that the corresponding z pixel will not be drawn.
  This function may only be used while the hardware is locked,
  except when indirect drawing is used.

INPUTS
  context - a context pointer
  x,y     - the starting position
  n       - number of pixels
  z       - pointer to a span of zpixels
  mask    - pointer to mask array. May be NULL

RESULT

EXAMPLE

NOTES
  This function is primarly intended for OpenGL implementations,
  which might need access to the Z buffer. This function
  is slow and should normally not be called.

  * IMPORTANT NOTE: *
  For speed reasons, this call is *NOT* compatible with indirect drawing.
  To use this call with indirect mode, you have to manually
            W3D_Flush
            ,
  and, should you use any drawing calls, you'll have to
            W3D_Flush
             again.

BUGS
  Indirect mode: the hardware is internally not locked for
  performance reasons, therefore the result might be wrong, if
  the corresponding buffer is swapped out.

SEE ALSO
```