Version 1.00

Creative Technology Extensions to the Glide® API

The following pages describe the extensions Creative has made to the Glide API to enable the advanced rendering features of newer cards in the graphics market. Specifically, these extensions address:

- 24-bit and 32-bit Color Rendering
- 24-bit and 32-bit Z-Buffer Depths
- Texture Patterns greater than 256x256 (up to 2Kx2K)
- Stencil Buffer Operations

All extensions documented below utilize the approved OpenGL® naming conventions for enhancements. Specifically, each interface call will have the company identifier “CTL” appended to each (i.e. grSetColorModeCTL).

Advanced Color Depth Interface

- void grSetColorPlanesCTL (GrColorPlanes_t);

This command is used to set the Color Depth that the Rendering System will use for the Color Planes. This command should be used after a call to grGlideInit and prior to a call to grSstWinOpen. By default, 16-bit will be used for rendering. The valid values for GrColorPlanes_t are:

GR_COLORPLANES_16	for 16-bit Rendering
GR_COLORPLANES_24	for 24-bit Rendering
GR_COLORPLANES_32	for 32-bit Rendering

- FxBool grVerifyColorPlanesCTL (GrColorPlanes_t);

This command is used to interrogate the system to determine which Color Depth modes are available on the given hardware. The user will pass in one of the valid GrColorPlanes_t definitions and the system will return a value of FXTRUE if the mode exists on the active hardware and FXFALSE if not.

- GrColorPlanes_t grInquireColorPlanesCTL (void);

This command is used to interrogate the system to determine which Color Depth mode is currently active on the hardware. The system will return one of the GrColorPlanes_t values to indicate the active mode.

In addition to these three new routines, existing routines that send or return color information (i.e. grLfbReadRegion) will return data in the proper format.

Advanced Z-Buffer Depth Interface

- void grSetDepthPlanesCTL (GrDepthPlanes_t);

This command is used to set the Z-Buffer Depth that the Rendering System will use for the Depth comparisons. This command should be used after a call to grGlideInit and prior to a call to grSstWinOpen. By default, 16-bit will be used for z-buffer operations. The valid values for GrDepthPlanes_t are:

GR_DEPTHPLANES_16	for 16-bit Z-Buffer Calculations
GR_DEPTHPLANES_24	for 24-bit Z-Buffer Calculations
GR_DEPTHPLANES_32	for 32-bit Z-Buffer Calculations

- FxBool grVerifyDepthPlanesCTL (GrDepthPlanes_t);

This command is used to interrogate the system to determine which Z-Buffer Depth modes are available on the given hardware. The user will pass in one of the valid GrDepthPlanes_t definitions and the system will return a value of FXTRUE if the mode exists on the active hardware and FXFALSE if not.

- GrDepthPlanes_t grInquireDepthPlanesCTL (void);

This command is used to interrogate the system to determine which Z-Buffer Depth mode is currently active on the hardware. The system will return one of the GrDepthPlanes_t values to indicate the active mode.

In addition to these three new routines, existing routines that send or return z-buffer information (i.e. grLfbReadRegion) will return data in the proper format.

Enhanced Texture Size Support

- GrLOD_t grInquireMaxTextureSizeCTL (void);

This command is used to interrogate the system to determine what is the size of the Maximum Sized texture pattern. 3Dfx chips are currently limited to 256x256 – but in newer chips, the size limit has increased to 2048 (2K). This function will return the Maximum Size Level of Detail (LOD) that can be supported. In addition, the following GrLOD_t definitions have been added.

GR_LOD_512
GR_LOD_1024
GR_LOD_2048

In addition to this inquiry function, all existing Glide routines that take a GrLOD_t variables (or variable included in a GrTexInfo or GrMipMapInfo structure) have been enhanced to support the larger sized textures.

Stencil Buffer Support

- void grRenderBufferCTL (GrBuffer_t);

This command has been enhanced to add the new GrBuffer_t values of GR_BUFFER_STENCILBUFFER. Upon executing this command, all subsequent drawing commands will be placed into the Stencil Buffer.

- FxBool grVerifyStencilPlanesCTL (void);

This command will simply return a value of FXTRUE if the hardware supports Stencil Planes and FXFALSE if not.

- void grStencilClearCTL (FxU32);

This command is used to clear the contents of the Stencil Buffer.

- void grStencilTestCTL (GrCmpFnc_t, FxU32);

This command takes a comparison function and reference value to use in the stencil test. The reference value is compared to the value currently in the stencil buffer using the comparison function. If the comparison fails, the operation defined in the Stencil Operation "fail" argument will be performed.

```
• void grStencilOperationCTL ( GrStencilOp_t fail,
                               GrStencilOp_t zfail,
                               GrStencilOp_t zpass );
```

This command specifies how the data in the stencil buffer will be modified when a pixel passes or fails the stencil test. The values of GrStencilOp_t are defined as:

```
GR_STENCILOP_KEEP
GR_STENCILO P_ZERO
GR_STENCILOP_REPLACE
GR_STENCILOP_INCR
GR_STENCILOP_INCRSAT
GR_STENCILOP_DECR
GR_STENCILOP_DECRSAT
GR_STENCILOP_INVERT
```

The fail argument is applied if the Stencil Test fails. If the Stencil Test passes, then zfail is applied if the Z-Buffer comparison fails, and zpass is applied if the Z-Buffer comparison is successful (or Depth Buffering is disabled).

In addition to these new commands, all existing routines to directly access the buffers (i.e. grLfbReadRegion) will operate properly on the Stencil Buffers.