cv3d

COLLABORATORS			
	TITLE : cv3d		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 24, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 cv3d **1**

1.1 The RtgMaster 3D Extensions V1.0 1

1.2 The RtgMaster 3D Extensions V1.0 1

1.3 The RtgMaster 3D Extensions V1.0 7

1.4 The RtgMaster 3D Extensions V1.0 8

1.5 The RtgMaster 3D Extensions V1.0 12

1.6 The RtgMaster 3D Extensions V1.0 14

1.7 The RtgMaster 3D Extensions V1.0 15

1.8 The RtgMaster 3D Extensions V1.0 15

Chapter 1

cv3d

1.1 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0"

Table of contents

Part A : An Introduction to modern 3D Accelerators

Part B : The rtgmaster 3D interface

Part C : The Virge - How do I use it

Part D : History

Part E : Authors and Copyright

Part F : The Speed of the Virge

Part G : Buglist

1.2 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions

An Introduction To Modern 3D Accelerators

As, following this tutorial, there will probably a lot of new concepts for a lot of
you. First: The Virge is a chipset, like others out, like AGA, maybe. There is no
"OS compatibility magic" around it, like some people tend to believe. There are some
differences between the Virge and AGA, though. It is very COMPLICATED to code a 3D
Chip in Hardware, because of this, you have to use the OS interface, if you do not
want to waste many weeks in learning how the chipset works. Also, most of the time,
those OS interfaces are done in a way, that they support several chipsets. (I will
try to make the rtgmaster 3D Extensions compatible to the CybervisionPPC and to the
Picasso IV 3D Extension also, when those two are released). Currently they only support
the Virge.

What does this mean for you, the programmer ? It means, that you have no longer
to hack the hardware, the rtgCV3D.library does this already for you (and for other

Chipsets other rtgmaster-sublibraries supporting the RtgMaster 3D Extensions will
appear).
It also means, that, if you, for example, want to create a 3D Engine, you won't need
any longer to waste CPU time with stuff like Texturemapping or Gouraud Shading...
the 3D Chipset will do that for you much faster, than the CPU can do it !!!
In the following section i will explain to you what exactly a modern 3D accelerator
like the Virge can do. In a later chapter you will learn how exactly this is done
with rtgCV3D.library. For now i only want to explain you the (partially) new concepts.

Texturemapping
==============

This is the basic stuff. Every 3D accelerator can do this. It involves putting a
texture on a Polygon. The Virge does this very fast, even faster than a 166 MHz
Pentium. Think over what this will mean for the Amiga... Contrary to faster Chips,
like the Permedia 2, though, the Virge only does the 3D Stuff very fast in quite
low resolutions, like 320x200, for example... in 800x600, this was the resolution
the original CV/3D 3D Demos ran in, it loses a lot of speed.
There are two sorts of Texturemapping, "without perspective-correction" and
"with perspective-correcttion". The problem is, most 3D Hardware involves texture
distortion, especially, if you turn around (look for example on the Game "Exhumed"
on the Sony Playstation for a fine example of "not-corrected" Texturemapping).
Not-Perspective correct Texturemapping is sometimes also called linear
Texturemapping. Please do NOT confuse this with Linear Texture-Filtering...
As you guess, not-corrected Texturemapping is faster. The Virge can do both,
not-corrected and corrected Texturemapping. The current rtgCV3D.library does not
yet support perspective-corrected Texturemapping, though. Also, perspective-corrected
Texturemapping is quite slow on the Virge (not on the Permedia-2, of course),
and it does not look THAT much better.

MIP-Mapping
===========

MIP-Mapping is a very special Texturemapping. You surely know the cases, when
you are very near a wall in your favourite Doom-Clone, and suddenly the pixels
get BIGGER AND BIGGER. It looks quite ugly, then. The problem is, that the game
engine only uses ONE texture for all different distances of the wall. The solution
is to use SEVERAL textures, one 128x128 texture, one 64x64 texture, one 32x32 texture,
and so on. The Virge chooses which of those textures would look best in the current
situation. This technique is called MIP-Mapping. Two games even do MIP-Mapping in
Software, this would be Quake and Trapped-2 (Demo). Of course, MIP-Mapping will
give you much more speed, if done in Hardware, like on the Virge. The current
rtgCV3D.library does not yet support MIP-Mapping.

Texture-Filtering

=================

Texture-Filtering is another method to make the textured scene look nicer, like
MIP-Mapping. Most 3D Chips (including the Virge) combine MIP-Mapping and Texture
Filtering to get the nicest image possible. Of course you can disable MIP-Mapping
and/or Texture Filtering for really slow scenes. To give you the general idea of
Texture-Filtering i am explaining you three different levels of texture display.

Level 1: Point-Sampling

Point-Sampling means that you take a pixel of the texture (also called a texel)
and put it on the polygon, directly. The Virge supports Point-Sampling combined
with MIP-Mapping, if you wish.

Level 2: Linear Texture-Filtering

Now you take 2 or 4 texels for every pixel to be drawn. To get the real texel
value, you INTERPOLATE between the texels. The Virge supports Texture-Filtering
with or without MIP-Mapping.

Level 3: Tri-Linear Texture-Filtering

Tri-Linear Texture-Filtering work with 8 texels, one of the CURRENT MIP-Map,
one of the NEXT MIP-Map. Of course this only works with MIP-Mapping. It is the
way the Nintendo 64 does the display, BTW. The Virge also supports this. Of course
Point-Sampling is faster...

As you probably see, a lot of Chips probably have a lot of different Filtering
Methods and making a program running on a lot of different 3D Chips is quite some
problem, because of this. A future version of the RtgMaster 3D Extensions will
support some sort of "ScreenMode System for Filtering Types" because of this.

Fogging

=======

Well, as you probably guessed, complicated scenes, maybe even displayed with
tri-linear MIP-Mapping, are slow (well, without 3D Hardware they would be
a slide-show, anyways :)) A way to speed up things is Fogging (the Nintendo
64 does this extensively, BTW). Fogging means that you interpolate between
the texel value and a certain fog color. This is normally done dependent of
the Z coordinate, in a way, that objects that are very far away are blurred
away in the fog and need NOT to be calculated. Nearer objects start blurring
a bit, to make the fog look still nice. Of course the Virge supports fogging,
but you can disable it, if you do not like it. The effects of fogging, if used
the way the Virge uses it (together with distance values) is also called
"depth cueing". On the Virge, Alpha Blending and Fogging cannot be used
at the same time. Fogging is not yet supported by the current version of
rtgCV3D.library.

Blending

========

Blending is a method to "mix" two textures. This can be used to lighten or
darken a scene. Do not confuse this with true Alpha-Blending. On the Virge
there are three sorts of Blending

- No Blending
- Complex reflection
- Decal reflection

In Complex reflection, current source pixel-color and texel-color get added, with
a maximum of 1. This lightens a scene. Decal reflection multiplies the colors,
darkening the scene. Palettized data can only use Decal reflection, at least on
the Virge.

Alpha-Blending

==============

Alpha-Blending is also a light effect. It can be used for transpareny effects.
The Idea is that you blend the Color of the new to be drawn pixel (at this
stage of color rendering) with the texture or with the source alpha value.
The smaller the value of alpha is, the higher the transparency. There are some
problems with Z-Buffering, that will be explained in the Z-Buffering section.
The Virge supports Alpha Blending. On the Virge, Fogging and Alpha Blending
cannot be used at the same time.

Z-Buffering

===========

The idea of Z-Buffering is solving the "Hidden Line Removal Problem". You know
this: You want to draw two polygons, but one is on top of the other. Now, if you
paint the one in front first, the back one should partially not be drawn at all.
Of course, if you always paint from back to front, there is no problem. Or you
have a different software solution, like for example sorting your polygons.
The solution the Virge offers, is a Z-Buffer. With a Z-Buffer you can specify a
certain depth, and all polygon parts, that are farer away than this Z value
will not be displayed (or all nearer or... there are a lot of sometimes not
very senseful combinations...).
For Z Buffering you need a Buffer of a size of x*y*2 Byte in Video RAM. Alternatively,
the Virge also offers the slower MUX-Buffering. Here the Z-Buffer can also be used
as Doublebuffering Buffer. MUX-Buffering is slower than Z-Buffering, though (and
Z-Buffering is already slow on the Virge... sorting your polygons with a software
solution is better here... the hardware support of Z-Buffering appearently is not
that great...).
You have to be careful, if you want to use Alpha Blending and Z-Buffering together.

If you want to do this, draw all opaque objects first. Then disable Z-Buffering and draw all transparent objects. When you disabled Z-Buffering and enabled Alpha Blending, always draw your objects from back to front.
About rtgCV3D.library support, well, here the Z-Buffering function does not work yet.

Gouraud-Shading

===============

The idea of Gouraud Shading is, that you sometimes do not even need textures to make 3D look fine. There are even some games that use Gouraud Shading, for example Virtua Fighter on the Sega Saturn Console. The Virge supports Gouraud Shading and is quite fast with it. It can do Gouraud Shading about 5 times faster than a 166 MHz Pentium (Note: I did not yet write a Benchmark on the Amiga for Gouraud Shading, so this is the value of a PCI Board on PC... as Zorro 2/3 are slower than PCI, the speed might be slower on Amiga... but on the other hand, all Amiga CPUs are MUCH slower than a 166 MHz Pentium, also, so there probably will be much speedup).
As to more speed information, have a look at the Chapter about it. On the Virge, Gouraud-Shading only really works, if you have a 15/16/24 Bit Color Mode (which is quite reasonable).

Wrap-Around

===========

Wrap-Around means, that the texture is wrapped around the object. That means, that if the texture is smaller than the object a certain Border-Color will be used to fill the rest of the object.

Lit-Texturing

=============

The idea of Lit-Texturing is to draw the textures with some lighting involved. The Virge supports this, both texture-corrected and not-texture-corrected. On the Virge, Lit-Texturing only really works, if you have a 15/16/24 Bit Color Mode (which is quite reasonable).

Polygon-Primitives

==================

The Virge supports textured Triangles and (of course not textured) lines in 3D.
The RtgMaster 3D Extensions do not yet support 3D Line-Drawing, but will so in the future.

Overlaying

==========

Overlaying is no 3D Feature, but the Virge supports it anyways. Instead of one main Buffer and one Double-Buffer, the Virge supports two Main-Buffers and two-Doublebuffers.
But why two Main-Buffers ? Well, the Hardware-Hackers among you might guess the

solution: It is for achieving a sort of DualPlayfield.
There is a certain operation:

```
if (Pixel_On_Second_Stream) Display(Pixel_On_FirstStream);
else Display(Pixel_On_Second_Stream);
```

and another Operation

```
if (Pixel_On_First_Stream) Display(Pixel_On_Second_Stream);
else Display(Pixel_On_First_Stream);
```

The special thing about this is, that there is no slowdown !!! Also, it is possible, then, to do a (partially) 24 Bit Display on an 8 Bit Display !!! Or vice versa...
Well, this is a very special Virge Feature, that probably will only exist on very few Boards... A future version of the RtgMaster 3D Extensions probably will support this feature.

The Concept of Memory
====================

As you probably noticed, with using 3D Chipsets, there are a lot of things that need memory :

- The Display and its Doublebuffered Map
- The Textures and MIP-Maps
- The Z-Buffer or MUX-Buffer

You see, there is a reason to do GFX Boards with 4 MB RAM.
The CV/3D has 4 MB RAM, for this. The CybervisionPPC will be released in a 4 MB and an 8 MB version.
I think the Picasso IV uses 4 MB, but not sure about it.
To calculate an example for the Virge/Cybervision/3D:
We do a 320x200 Screen. We also use Overlaying and Doublebuffering. We use 15 Bit.
(Well, the Virge is still quite fast in 15 Bit).
320x200x15x4 = 512 KB
Workbench Screen 1024x768 16 Bit (i do not think anybody uses a bigger one): 1.5 MB
1 MIP-Map 128x128+64x64+32x32+16x16 = 42.5 KB
1 Z-Buffer 320x200 = 128 KB
So we can fill the rest of the Memory without reloading textures with:
45 such MIP-Maps. (on an 8 MB CybervisionPPC we could use 90 of them...)
Well, should be enough for a game :)
Of course you can dynamically load new textures, but be warned: Uploading textures to the Board goes through the Zorro-Bus, and is SLOW. Painting the Textures does not involve the Zorro-Bus very much (you give the command through the bus, and the Virge completely does the rest), but you only should upload textures, when this is NOT time-critical. But well, 45 of them should be enough... when you deal with 8 Bit textures you even have 90 of them ... :)
To optimize Texture handling, there are some options:

- Reload Textures in not-time-critical parts (for example, when a new level is loaded)
- Do only the monsters in true 16 Bit, the dungeon in 8 Bit, or vice versa

1.3 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions

The Rtgmaster 3D Interface

To understand the following you should be already acknowledged with the rtgmaster
interface. rtgmaster is an API for :

- Direct Access to the Video RAM of GFX Boards
- GFX Board Compatible Coding of Games which still run on ECS/AGA
- Fast Chunky-Copy functions
- Some special functions like Text, Fonts,...

These days there are some new Boards in the making that support 3D in Hardware:

- The Cybervision 3D with the 3D Virge Chip (already available)
- The Picasso IV with an announced 3D Extension (not yet available)
- The CybervisionPPC with the Permedia-2 Chip (not yet available)

It makes sense to add 3D Support to rtgmaster. The problem, of course, is,
3D Chips are in no way standardized. So it is difficult to support all 3D Chips
with one API. Additionally, code that uses the 3D Chips won't run on Boards
that do not have 3D Chips, like for example the Piccolo SD64 or the Picasso II.

I decided the following for the future of rtgmaster:

- You can always call rtgmaster.library functions. These will run on all
rtgmaster-compatible Boards.
- Additionally you can call some functions directly from a rtgmaster sublibrary.
Functions inside a sublibrary won't run on all Boards.

To access the Library Base of a sublibrary, you have to access the Chipset3D
structure that you got returned using GetRtgScreenData. Inside this structure
you will find the Library Base of the 3D Extension.

You have always to use the "special 3D version" of the sublibrary (for example
rtgCV3D.library instead of rtgCGX.library). Note also, that GetRtgScreenData
with the Tag grd_3DChipset also gives you an overview about the features provided
by a special 3D Chip. Not all of them offer the same...

It is INTENTIONALLY, that i did not put the 3D Functions into rtgmaster itself.
I wanted to make it ABSOLUTELY CLEAR, that these functions are not available
on every Board, and that your program only will run on 20% of the existing
Boards, if you use them. Of course you could use them "optionally" (for example,
if your texturemapper already works with triangles, why not use the PaintT3D
function when a Virge was found, instead of the software solution ?)

1.4 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0

The Virge - How to use it

The first call you should do, when using the rtgmaster 3D extensions (after you
got the LibBase for the Sublibrary your screen opened on) is a call of the
standard rtgmaster function GetRtgScreenData, with the tag grd_3DChipset.
It is urgent, that the ti_Data field is really set to 0.
If it is still 0 after the call, then this sublibrary, on which the screen
opened, does not support the rtgmaster 3D Extensions. If it returned something
DIFFERENT from 0, then this is a pointer to a Chipset3D structure.
Note: This structure really shows you the features of a certain SUBLIBRARY,
not of a certain Chip, so you know that all features shown in this structure are
really supported by the sublibrary.

```
struct Chipset3D
{
struct Library *Base3D;
int mapping;
int p_mapping;
int filters;
int tmapclr;
int fog;
int blend;
int ablend;
int zbuf;
int mux;
int gouraud;
int lit;
int line;
int wrap;
int ovl;
int mem;
int size;
int lsize;
int flags;
}
```

Base3D is the Library Base of the Sublibrary. This is the ONLY legal way to get
this Library Base.
If mapping is 1, this Chip supports non-perspective-correct texture-mapping, if it is 0,

it does not support it.

p_mapping does the same for perspective-correct texture-mapping.

filters lists supported texturefilters for a Board. Each set bit means one supported

Filter Type.

tmapclr lists supported texture color formats. One Bit means one supported format.

fog, blend and abland are flags, if fogging, blending and/or Alpha-Blending are

supported.

zbuf, mux and gouraud do the same for Z-Buffering, MUX-Buffering and Gouraud-Shading.

lit, line, wrap and ovl do the same for Lit-Texturing, 3D-line-Drawing and Overlay-Functions.

mem is the size of the texture memory in Bytes (changed since first version !!!)

size is the max. size of non-linear mapped textures (128 means 128x128, for example)

lsize is the max. size of linear mapped textures (linear mapped = not-perspective

correct)

flags indicate some special conditions that might be true for future Chipset.

The structure might be expanded in the future (Always Downward Compatible !!!)

filters are:

0 TPPM1 (Mip-Mapping, no Filtering)

1 TPPM2 (Mip-Mapping, linear Filtering)

2 TPPM4 (Mip-Mapping, Bi-Linear Filtering)

3 TPPM8 (Tri-Linear MIP-Mapping)

4 TPP1 (Point-Sampling)

5 TPPV2 (For YUV-Format...)

6 TPP4 (Bi-Linear Filtering)

Color Types are :

0 CF_ARGB32

1 CF_ARGB16

2 CF_ARGB15

3 CF_ALP4BLE4

4 CF_BLEND4LO

5 CF_BLEND4HI

6 CF_LUT8

7 CF_YUYV

Be careful !!! These constants changed since the last version. Now ARGB32,LUT8...

are the constants for smr_ChunkySupport, and CF_... the ones for the 3D Expansion

(i choose to change the name of the 3D Expansion Constants, as the 3D Expansion

is not yet released, anyways, and smr_ChunkySupport Constants should stay downward

compatible).

Current flags are :

0 Fogging and Alpha-Blending cannot both happen

1 Alpha-Blended Objects can cause problems with Z-Buffering

2 Wrap-Around is not supported for p-correct textures

3 In Palettized Mode only Decal Blending is possible

4 Gouraud Shading does not work in Palettized Mode (Reasonable :))

5 Lit-Texturing does not work in Palettized Mode (Reasonable :))

What you make of all your data, is your stuff. But it is recommended checking it at
startup of your program. Currently this is not much of a problem, as the CV/3D is
the only 3D Board available for the Amiga, up to now. But for future compatibility...

Now as to the actual functions...

there are in fact 9 of them:

- ClrZBuf clears the Z Buffer/MUX Buffer. It takes the RtgScreen as parameter in a0.

Note: rtgCV3D.library does not support the more exotic Z-Buffer configurations. It
sets the configuration of the Chip correctly for the usual Hidden-Line-Stuff.

- CreateTex creates a Texture Handle (see below)
- DeleteTex frees the handle again
- LoadTex loads a Texture to memory (see below)
- FreeTex frees the Texture again
- GetTexAttr examines the attributes currently set in the 3D Chip (see below)
- SetTexAttr sets a certain attribute of the 3D Chip to a certain value (see below)
- PaintT3D paints a textured or shaded triangle to the screen (see below)
- ClearBuf clears a Backbuffer (does not work yet in the Virge
Implementation), getting the RtgScreen in a0, the Buffer Number (0 or 1) in d0.

There might be more functions in the future.

CreateTex takes the RtgScreen in a0 und the TagList in a1 as parameters.

If this function finds something not supported, it tries to make the best out
of it. But better do not use something not supported.

Currently supported Tags (more in the future) :

tex_TexMap : Pointer to the Texture

tex_MapSize : Size of the Map

tex_ColorFmt : Texture Format used in the texture

tex_FilterType: Filter Type to be used

tex_TexWrap: Flag, if Wrap-Around should be used

tex_LitTex: Flag, if Lit-Texturing should be used

tex_AlphaBlend: Alpha Blending Type (or none) to be used

Currently supported Filters in rtgCV3D.library :

4 TPP1

5 TPPV2

6 TPP4

As to the current version, no perspective-correction is done. A later version will support perspective-correction, of course.

Supported Colorformats in rtgCV3D.library :

0 CF_ARGB32

1 CF_ARGB16 ;%aaaarrrr ggggbbbb

2 CF_ARGB15 ;%arrrrrgg gggbbbbb

3 CF_ALP4BLE4

4 CF_BLEND4LO

5 CF_BLEND4HI

6 CF_LUT8

7 CF_YUYV

Supported Alpha Blending Types in rtgCV3D.library :

-1 ABlend_Source

0 ABlend_None

1 ABlend_Texture

Supported Blending Types in rtgCV3D.library :

0 Blend_Com

1 Blend_Mod

2 Blend_Decal

GetTexAttr can be used to find out to what value one of the tex_ Attributes was set. You specify the RtgScreen in a0, the TexHandle to examine in a1, the Attribute (for example tex_MapSize) in d0, and get the value of the examined Attribute back in d0.

As to SetTexHandle, you specify the RtgScreen in a0, the TexHandle in a1, the Attribute Number (for example tex_MapSize) in d0, the intended value in d1.

It is also possible to change the texture data applied to a TexHandle. Of course this involves large amounts of data being sent over the Zorro Bus. If you want to do this, use LoadTex, providing the RtgScreen in a0, the TexHandle to modify in a1, and the filename of the Texture File in a2. If you loaded a texture this way, you have to free the allocated memory using FreeTex again, providing the RtgScreen in a0 and the TexHandle in a1.

The last and most important call is PaintT3D. It is used to paint a triangle (which can be shaded or textured) onto the screen, using the Virge (or other) hardware. You have to specify three parameters: RtgScreen in a0, Triangle Data in A1 and BlendMode in D0.

The BlendMode is just the above specified BlendMode, Blend_Comp or Blend_Decal.

The Triangle structure looks like that:

```
typedef struct
{
Point3D p1,p2,p3;
```

struct TexHandle *th;

} Triangle3D;

p1,p2 and p3 are the three 3D-points of the triangle, th is the Texture to be

used with it. Point3D is defined as :

typedef struct

{

LONG x,y,z,u,v;

ULONG color;

} Point3D;

x,y,z are the coordinates in Polygon Space, u and v are the coordinates in

Texture Space.

The color value is used if you do Gouraud-Shading or Lit-Texturing.

As to the Virge implementation, the next planned new features of rtgCV3D.library are:

- Making it independent of cgx3dvirgin.library
- Implementing Fogging
- Implementing a Virge-Native Doublebuffering
- Implementing a better RtgWaitTOF() for the Virge
- Autodocs

After that i will implement:

- perspective-correct Texturemapping
- MIP-Mapping
- Overlay-Stuff
- more Texture-Filters
- Adding a new function to the 3D interface for 3D Linedrawing

And finally:

- Adding Virge-Native Blitting and 2D Linedrawing
- Doing a PowerPC Native Version of the whole thing
- Doing versions for CybervisionPPC and for the 3D Expansion of the Picasso IV

(well, when i am finished THIS FAR, they probably will be released... :)

This is a LOT of work...)

- Some more examples

1.5 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0

Speedtests

I did not yet speedtest the Virge-Chip on the Cybervision/3D (not yet had the time

for this), but i already got some Benchmarks from a PC guy. The benchmarks were

done on a 166 MHz Pentium, using a Virge based PCI-GFX-Board.

You have to consider, when you have a look at the results, that the Zorro Bus is slower than PCI, but also, a 68040 is slower than a 166 MHz Pentium. So i think the Virge will cause wonders, still. Remember, after you have loaded the textures to the Video RAM the only things that cross the Zorro Bus are some polygon coordinates (which are much fewer than if you put the textures to Video RAM, like you have when you do without texturemapping hardware).

```
+-----------------------------------------------------------------+
| |
| Virge 320x200x15bpp |
| ------------------- |
| |
| Gouraud: 210 |
| |
| Point-sampled textures |
| |
| linear persp |
| flat 128 72 |
| RGB 93 64 |
| |
| Bilinear-filtered textures |
| |
| linear persp |
| flat 75 68 |
| RGB 59 57 |
| |
+-----------------------------------------------------------------+
| |
| Virge 640x240x15bpp |
| ------------------- |
| |
| Gouraud: 128 |
| |
| Point-sampled textures |
| |
| linear persp |
| flat 73 35 |
| RGB 49 31 |
| |
| Bilinear-filtered textures |
| |
| linear persp |
| flat 34 17 |
| RGB 28 16 |
| |
+---------------------------------------------------------------+
| |
| Software rendering 320x200x15bpp |
| -------------------------------- |
| |
| Gouraud: 41 |
| |
| Point-sampled textures, n-sided polygons rasterizer |
| |
| linear persp |
| flat 31 29 |
| RGB 27 26 |
| |
| Point-sampled textures, triangle rasterizer (tesselating) |
| |
| linear persp |
| flat 31 28 |
| RGB 25 23 |
| |
+---------------------------------------------------------------+
```

1.6 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0
Authors and Copyright
Most of the currently existing code is based on cgx3dvirgin.library which was coded
by Gerd Kautzmann and Frank Mariak. Many thanks to them for their support and to
the permission of Frank Mariak to use cgx3dvirgin.library for a independent games
API (Frank is a busy man and does not have much time for the 3D API, sadly, this was
the cause why till now cgx3dvirgin.library was not released to the public... Frank
decided because of this supporting me, so that i can do a 3D API for Amiga...)
BTW, in the original 3D Demos (which i remember as quite slow) the keyboard function
consumed about 90% of the time... this is of course fixed in rtgCV3D.library...
well the 3D Functions in cgx3dvirgin.library where in fact a really good job, but
the Keyboard functions ? Not.

1.7 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0
Bugs in the Virge Implementation
- The Virge Detection does not work correctly. It tries Virge functions even on a
not-3D Board.
- The Screenmode Save function does not work correctly. If you save the Screenmode,
the rtgCGX.library Screenmode will be started, instead of the rtgCV3D.library
Screenmode.
- Clearing the Backbuffer always causes flickering. I disabled it in the demo
because of this. I still have to find a way to do this in working way. If anybody
has an idea of what goes wrong, please email me. 3D Support NEEDS Backbuffer
clearing and Doublebuffering !!!
- RtgGetMsg is uncomplete. im_Seconds and im_Micros are not yet set correctly,
Right and Middle Mousebutton are not yet handled.
- Z Buffering does not yet work
- LoadTex and FreeTex do not yet work, do this manually, up to now !!!
- There seems to be something strange inside the library. After you ran a program
once, the Rtg-Screenmode-Requester won't work until you reboot, also sometimes
the library crashes at startup (not sure, if this is a bug in my own code or in
cgx3dvirgin.library, which also is quite ... unfinished)

1.8 The RtgMaster 3D Extensions V1.0

The RtgMaster 3D Extensions V1.0
History
1.0 rtgCV3D.library created. It is a interface between rtgmaster (which supports
Direct Video RAM Access, Text, Mousepointer Change, Linedrawing... but no
3D) and cgx3dvirgin.library (which supports 3D, but none of the other
features). Thanks to Frank Mariak for providing cgx3dvirgin.library.
Future versions will be independent of cgx3dvirgin.library (and will
have expanded functionality)
2.41 Tons of internal changes, still it sometimes crashes the system at startup.
RtgGetMsg was updated a lot and a lot of bugs are fixed. Screenmode name
is again CV/3D: in this current version, might change again in the future
(due to internal changes). Should still be considered an Alpha Version.
2.42 Mainly RtgGetMsg is updated, bugfix in RtgInitRDCMP, some minor fixes
asides from that.
2.43 RtgGetMsg again OS conform, but in a bit a different way. Now it is

acceptable :) Also support for ClearBuf added, so backbuffer clearing now
works. Some small bugfixes. Now the lib works without crashes, and
also a bug in OpenRtgScreen of the lib was fixed. Now we soon move from
Alpha Version to Beta Version :)
26.0 Changed version number to conform with rtgmaster.library version number.
Changed Color Format constants to CF_... (the constants without CF_...
are used in smr_ChunkySupport already !!!). CreateTex now returns 0,
if not enough texture memory is available. Changed Chipset3D->mem to
"number of Bytes in Texture-Memory".