相关主题 RECOMMEND ARTICLE
- HDR渲染器的实现(基于OpenGL)
- Vertex Cache (OpenGL实现)
- Slope(斜坡) 法线生成算法，在地形渲染中的应用
- 简单的运动模糊效果实现
- 具有真实感的3D图形(下)
- 具有真实感的3D图形(上)
- 多动画集在D3D下的渲染
- Portal系统

MORE

推荐文章 RECOMMEND ARTICLE
- 数据广播方案的优化
- 网络游戏的位置同步
- 游戏音乐制作案例之《战火 红色警戒》音效制作揭秘
- 英雄连Online 原画
- 游戏音乐制作案例之《乱武天下》
- 游戏音乐制作案例之《诛仙》
- 《鹿鼎记》最新原画
- MIDP2.1规范的新特性

MORE

热门文章 HOT ARTICLE
- [电子书下载]游戏设计 — 原理与实践
- [电子书下载]网络游戏开发
- 游戏设计全过程
- [电子书下载]游戏设计技术
- [电子书下载]游戏设计理论
- CS游戏人物模型制作教程
- CG人物插画基本流程
- [转贴]MAX高级人头教程

MORE

您的位置： 图形技术

文章标题	地形教程 - 一个TGA库		
来源：	[Azure]	浏览：	[765]

Terrain Tutorial

A TGA library
一个TGA库

This library is introduced here in order to make this tutorial self-contained. The library is pretty simple, handling only uncompressed images, RGBA or greyscale. No support for color palettes is provided. If you're interested in a full-blown TGA library I would suggest a look at Paul Groves' TGA loader. Check out his home page.
http://home.clara.net/paulyg
因为这个教程使用到了这个库，所以我们引入它近来。这个库相当的简练，只能处理非压缩的图像，RGBA或者灰度图。不支持调色板。如果你对功能全面的TGA感兴趣，我建议你看下 Paul Groves' TGA loader，看看下面这个地址。

The functions in this library include loading and saving TGA images, colour reduction from RGB to greyscale, and a screen grabber.
这个库的函数包含读取和存储TGA图像，灰度转换,一个屏幕抢夺(?)功能.

TGA file structure
TGA文件结构

A TGA file has a header that consists of 12 fields. These are:
一个TGA的文件头包含12个域,它们是:
id (unsigned char)
colour map type (unsigned char) 色彩图类型
image type (unsigned char) 图像类型
colour map first entry (short int) 颜色图第一个入口
colour map length (short int) 颜色图长度
map entry size (short int) 图入口尺寸
horizontal origin (short int) 水平起始
vertical origin (short int) 垂直起始
width (short int) 宽度
height (short int) 高度
pixel depth (unsigned char)像素深度
image descriptor (unsigned char) 图像描述
From all these fields we only really care about the image type, in order to find out if the image is uncompressed and it is not color indexed, the width and height of the image, the pixel depth, and finally the image descriptor that contains the image pixels.
全部的这些域里面,我们唯一关心的只是图像类型(image type),为了找出图是非压缩的并且颜色不是索引的,和图像的高度和宽度,颜色深度,最后图像表述和图像的像素内容.

Some possible values for the image type are:
一些可能的图像类型如下:
1 - colour map image
2 - RGB(A) uncompressed
3 - greyscale uncompressed
9 - greyscale RLE (compressed)
10 - RGB(A) RLE (compressed)
The only types that we're dealing with here are 2 and 3. As for the pixel depth, it represents the number of bits per pixel used, i.e. a greyscale image has 8 for pixel depth, where as a RGBA has 32.
我们只能处理2,3两种图像类型,对于像素深度,它表达了每个像素使用的字节数,例如,灰度图每个像素使用8位,而RGBA每个像素使用32位.

One note of interest is that a TGA stores the pixels in BGR mode, i.e. the red and blue components are swapped, relative to RGB. This implies that we'll have to swap them when we load or save the image.
这里要提醒的是TGA储存像素的方式是BGR模式，相对于RGB模式，红色和蓝色分量互换了位置。这个也预示我们在读取和储存的时候，我们需要交换它们的位置。

Now for some library details. The following status codes where defined:
下面定义了一些状态吗:
TGA_ERROR_FILE_OPEN
TGA_ERROR_READING_FILE
TGA_ERROR_INDEXED_COLOR - when we're presented with a color indexed file
TGA_ERROR_MEMORY
TGA_ERROR_COMPRESSED_FILE - when we're presented with a compressed file
TGA_OK - This is what we want! 这才是我们所需要的
The following structure provides the necessary fields to hold the image information and pixels:
下面的结构提供了必要的域来保存图像信息和像素:

```
typedef struct {
int status;
unsigned char type, pixelDepth;
short int width, height;
unsigned char *imageData;
}tgaInfo;
```

The following functions are available in the library:
下面这些函数是库中可用的：

```
tgaInfo* tgaLoad(char *filename);
```

Parameters:
filename - the name of the image file
文件名 - 图像的名称

This functions returns a structure with all the image info and pixels. Pixels are stored in imageData, which is an one-dimensional array with values from 0 to 255. In order to read this array correctly, we'll need to check the values of pixelDepth, width and height. The value of pixelDepth tells us which information is in the array. There are 3 possible values with the following meaning:
这个函数返回一个结构有着图像信息和像素信息。像素数据储存在imageData里，它是一个值域从0到255的一维数组。为了正确的解读这个数组，我们需要检查pixelDepth值，宽度和高度。值pixelDepth告诉我们他们是下面三个可能值之一：
8 - The image is greyscale, each array value corresponds to a pixels intensity. 8 - 图像是灰度的，数组的每个值代表相应的像素强度。

24 - The image is RGB, each pixel requires 3 components of the array. The array should look like R,G,B,R,G,B,... 24 - 图像是RGB格式，每个像素需要3个数组成份，数组应该是这样排列R，G，B，R，G，B

32 - An RGBA image, where each pixel requires 4 components. The array looks like R,G,B,A,R,G,B,A,... 32 - 图像是RGBA格式，每个像素需要3个数组成份，数组应该是这样排列R，G，B，A，R，G，B，A
When calling this function is it always a good idea to check the value of the fieldstatus to see if there were any errors.
当我们使用这个函数的时候，一个好的习惯是检查fieldstatus看看有没有什么错误。

The next function allows us to save pixels as a TGA image
下一个函数允许我们把像素储存成TGA图像

```
int tgaSave(char *filename, short int width, short int height, unsigned char pixelDepth, unsigned char *imageData);
```

Parameters:
filename - The file name where we want to save the image (extension is required!).
width - The width of the image
height - The height of the image
pixelDepth - The number of bits per pixel, 8 for greyscale, 24 for RGB, and 32 for RGBA
imageData - The image pixels

参数:
filename - 我们需要储存的文件名（需要扩展名）
width - 图像的宽度
height - 图像的高度
pixelDepth - 每个像素占用的位数，8是灰度图，24是RGB格式，32时RGBA格式
imageData - 像素数据
This function returns a status code. TGA_OK means everything went smoothly.
函数返回status码， TGA_OK说明一切进展的很顺利。

The next function is provided so that we can save series of images. For instance, if the file name is "foo" then the first time this function is called it will save an image as "foo0.tga", the second time "foo1.tga", and so on.
下一个函数提供了保存一系列图像的功能，例如，如果文件名是“foo“然后第一次函数调用时保存的文件名是"foo0.tga“，然后第二次就是"foo1.tga“，以此类推。

```
int tgaSaveSeries(char *filename, short int width, short int height, unsigned char pixelDepth, unsigned char *imageData);
```

Parameters:
filename - The file name where we want to save the image (no extensions in here please).
width - The width of the image
height - The height of the image
pixelDepth - The number of bits per pixel, 8 for greyscale, 24 for RGB, and 32 for RGBA
imageData - The image pixels
As always, it is a good idea to check the return value of the function to see if the operation was completed successfully. The next function allows you to take a screen shot and save it to a TGA image. With this function it is possible to grab the entire viewport, or just a part of it. The syntax is as follows:
像平常一样，好的习惯是检查函数的返回值看看操作是否成功。下一个函数让我们可以抓取屏幕再存储成TGA图像。这个函数可能占有整个视口，或则它的一部份。语法如下：

```
int tgaGrabScreenSeries(char *filename, int x,int y, int width, int height);
```

Parameters:
filename - the image file name 文件名
x - The x component of the lower left corner of the image.图像左下角的x分量
y - The y component of the lower left corner of the image. 图像左下角的y分量
width - The width of the image. 图像的宽度
height - The height of the image. 图像的高度
The following code snipet illutrates how to capture the whole viewport (assuming a viewport from 0,0 to w,h) , and only its top half.
下面的代码段展示了如果抓取整个视口（假设视口是从0,0到w,h），和只有一半的高度宽度。

```
tgaGrabSeries("bla", 0,0,w,h);
tgaGrabSeries("bla", 0,h/2,w,h/2);
```

The code for the above function is:
上面的函数的代码如下：

```
int tgaGrabScreenSeries(char *filename,
int xmin,int ymin,
int xmax, int ymax) {

int w, h;
unsigned char *imageData;
```

```
w = xmax - xmin;
h = ymax - ymin;

imageData = (unsigned char *)malloc(
sizeof(unsigned char)
* w * h * 4);

glReadPixels(xmin,ymin,xmax,ymax,
GL_RGBA,GL_UNSIGNED_BYTE, (GLvoid *)imageData);

return(tgaSaveSeries(filename,w,h,32,imageData));
}
```

The key is the OpenGL function glReadPixels. This function reads a rectangular area from the frame buffer and stores it in an array (this array must have previously allocated memory). This function has the following syntax:
这个函数的关键是OpenGL的函数glReadPixels，这个函数从帧缓存中读取一个矩形,并且把它储存在一个数组里（数据必须是预先分配了内存的），这个函数的语法如下：

```
void glReadPixels(GLint x, GLint y, GLsizei width, GLsizei height, GLenum format, GLenum type, GLvoid
*pixels);
```

Parameters:
x - the x component of the lower left corner of the requested area 区域左下角坐标的x分量
y - the y component of the lower left corner of the requested area 区域左下角坐标的y分量
width - the width in pixels of the area 区域的像素宽度
height - the height in pixels of the area 区域的像素高度
format - determines which data we're going to read. See bellow for some possible values.像素格式，可能是下面的可能值
type - the data type of the data we're going to read. 我们准备读的数据类型
pixels - An array with the pixel information. The array data depends on the format and type. 数据首地址
The format specifies if we're getting RGBA data, or just the Green component. Some possible values are:
下面是像素格式的定义，一些可能的值如下：
GL_RGB - Read the three color components
GL_RGBA - RGB + alpha channel
GL_GREEN - Read only the green component
GL_BGR, GL_BGRA - This format reads the same information as GL_RGB, and GL_RGBA, but returns the Blue component first instead of the Red one.
NBote that GL_BGR and GL_BGRA were only introduced in OpenGL 1.2. As for type the value we want is:
GL_UNSIGNED_BYTE - each component is represented has a number between 0 and 255
This is the data type of each component in a TGA image.
这个是TGA图像每个部分的数据类型。

This next function converts a RGB image to a greyscale image. The formula used was posted on the openGL.org discussion groups some time ago.
下一个函数将RGB图像转换成了灰度图。OpenGL.org 的讨论组以前贴过这个公式。
greyscale = 0.30 * R + 0.59 * G + 0.11 * B
This function returns the modified image information. Besides imageData, the fields pixelDepth and type are also altered accordingly to reflect the new image type.
这个函数返回修改后的图象信息，除了imageData, pixelDepth域，其他的都变成了新的图像信息了。

```
void tgaRGBtogreyscale(tgaInfo *info);
```

Parameters:
This last function releases the memory, so when using the image for a texture or a height map, the image can be destroyed afterwards.
最后一个函数释放了内存，所以当使用一个纹理或者高度图后，图像的数据就可以被释放掉了。

```
void tgaDestroy(tgaInfo *info);
```

Parameters:
The next section presents the fully commented code for this library.
下个章节，我们将放出全面注释的库源代码。