河北师范大学软件学院
Software College of Hebei Normal University

Android源生AR应用开发

第五讲 纹理贴图

智能设备教研室

目录

纹理贴图

- **纹理** 是表示物体表面细节的一幅或几幅二维图形，也称纹理贴图（Texture Mapping），当把纹理按照特定的方式映射到物体表面上的时候能使物体看上去更加真实。纹理映射是一种允许我们为三角形赋予图象数据的技术，能够更细腻更真实地表现场景。

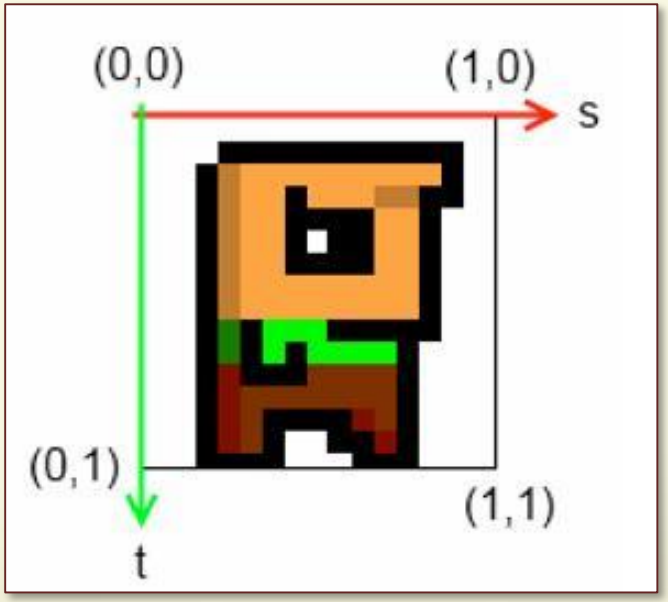
纹理贴图

❖纹理坐标

- 纹理坐标在x和y轴上，范围为0到1之间（当然也可以大于1）。使用纹理坐标获取纹理颜色叫做采样(Sampling)。纹理坐标起始于(0, 0)，也就是纹理图片的左上角，终始于(1, 1)，即纹理图片的右下角。
- 它的坐标系是作(s, t)，有时也写作(u, v)。

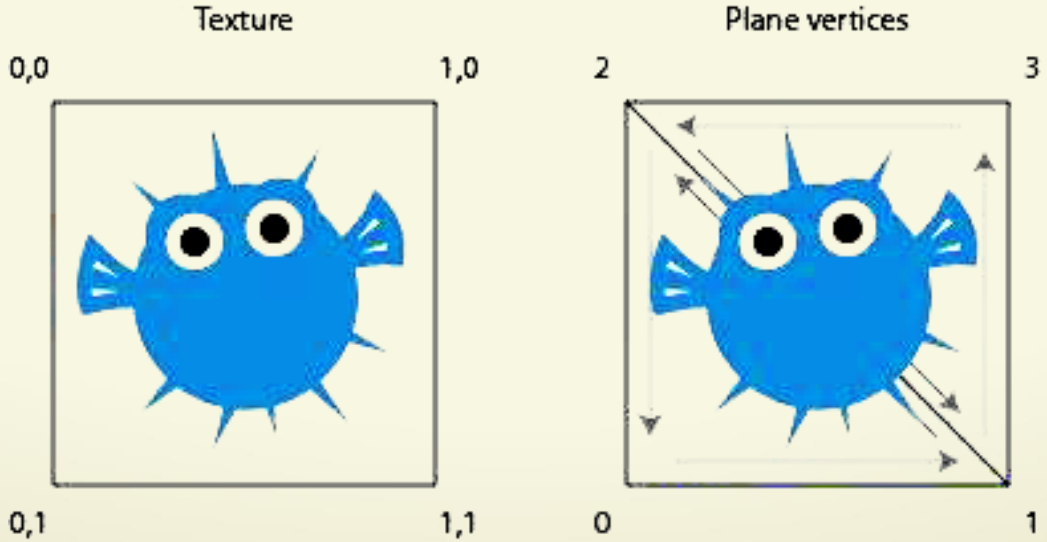
纹理贴图

- OpenGL要求纹理的高度和宽度都必须是2的n次方大小，只有满足这个条件，这个纹理图片才是有效的。



纹理贴图

- 纹理坐标 和 顶点坐标映射关系
  - ✓ 顶点顺序： 2 0 1 2 1 3
  - ✓ 纹理坐标： 0,0 0,1 1,1 0,0 1,1 1,0



纹理贴图

❖纹理环绕方式

- 纹理坐标的范围通常是从(0, 0)到(1, 1)。但是如果纹理坐标不在该范围里，OpenGL ES默认的行为是重复这个纹理图像，也可以自己设置其它处理的方式。

环绕方式(Wrapping)	描述
GL_REPEAT	对纹理的**默认**行为，重复纹理图像。
GL_MIRRORED_REPEAT	和GL_REPEAT一样，但每次重复图片是镜像放置的。
GL_CLAMP_TO_EDGE	纹理坐标会被约束在0到1之间，超出的部分会重复纹理坐标的边缘，产生一种边缘被拉伸的效果。
GL_CLAMP_TO_BORDER	超出的坐标为用户指定的边缘颜色。

纹理贴图

GL_REPEAT

GL_MIRRORED_REPEAT

GL_CLAMP_TO_EDGE

GL_CLAMP_TO_BORDER

纹理贴图

- 可以使用glTexParameterf()函数对单独的一个坐标轴设置（二维纹理为s、t坐标，三维纹理为s、t、r坐标）

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
                GL_CLAMP_TO_EDGE);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,
                GL_CLAMP_TO_EDGE);
```

  - 参数 target ：纹理目标是
  - 参数 pname ：指定坐标轴S轴、T轴、R轴
  - 参数 param ：环绕方式

纹理贴图

- 如果选择 GL_CLAMP_TO_BORDER 选项，还需要指定一个边缘的颜色。这需要使用glTexParameterfv()函数，用GL_TEXTURE_BORDER_COLOR作为它的选项，并且传递一个float数组作为边缘的颜色值

```
float borderColor[] = { 0.5f, 0.5f, 0.5f, 1.0f };
glTexParameterfv(GL_TEXTURE_2D,
                 GL_TEXTURE_BORDER_COLOR,
                 borderColor);
```

纹理贴图

❖纹理过滤

- 纹理坐标不依赖于分辨率，它可以是任意浮点值，所以OpenGL ES需要知道怎样将纹理像素映射到纹理坐标。OpenGL ES默认的纹理过滤方式是邻近过滤。

线性过滤 GL_LINEAR

邻近过滤 GL_NEAREST

纹理贴图

纹理过滤	描述	总结
GL_LINEAR 线性过滤	它会基于纹理坐标附近的纹理像素，计算出一个插值，近似出这些纹理像素之间的颜色。一个纹理像素的中心距离纹理坐标越近，那么这个纹理像素的颜色对最终的样本颜色的贡献越大	GL_LINEAR能够产生更平滑的图案，很难看出单个的纹理像素。
GL_NEAREST 邻近过滤	当设置为GL_NEAREST的时候，会选择中心点最接近纹理坐标的那个像素。	GL_NEAREST产生了颗粒状的图案，我们能够清晰看到组成纹理的像素

纹理贴图

- 使用glTexParameterf()函数为放大和缩小指定过滤方式。

```
glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER,
                GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MAG_FILTER,
                GL_LINEAR);
```

纹理贴图

❖纹理生成

1. 创建纹理对象

```
void glGenTextures(int n,
                   int[] texture,
                   int offset);
```

- 参数 n ：表示需要创建纹理对象的个数
- 参数 texture ：用于存储创建好的纹理对象句柄
- 参数 offset ：偏移量

纹理贴图

2. 将纹理对象设置为当前纹理对象

操作纹理，传入纹理句柄作为参数，每次Bind之后，后续操作的纹理都是该纹理。

```
void glBindTexture(int target, int texture);
```

- 参数 target ：指定绑定的目标，GL_TEXTURE_2D
- 参数 texture ：纹理对象句柄

纹理贴图

3. 通过glTexParameterf()函数指定纹理格式。这里包括纹理横向和纵向的重复方式和纹理在放大和缩小（同样纹理离远和离近）时的处理，即纹理环绕方式和纹理过滤。
    - GL_TEXTURE_WRAP_S
    - GL_TEXTURE_WRAP_T
    - GL_TEXTURE_MAG_FILTER
    - GL_TEXTURE_MIN_FILTER

纹理贴图

4. 给纹理传入图像数据，可以使用GLUtils.texImage2D()方法，或者glTexImage2D()方法。

```
void GLUtils.texImage2D (int target, int level,
                         Bitmap bitmap, int border)
```

- 参数 target ：操作的目标类型，GL_TEXTURE_2D
- 参数 level ：纹理的级别，0表示基本图像层
- 参数 bitmap ：纹理图像
- 参数 border ：纹理边框尺寸

纹理贴图

```
void glTexImage2D(int target, int level,
                  int components, int width,
                  int height, int border,
                  int format, int type,
                  Buffer  pixels);
```

- 参数 target ：操作的目标类型，GL_TEXTURE_2D
- 参数 level ：纹理的级别，0表示基本图像层
- 参数 components ：GLES20.GL_RGBA
- 参数 width、height ：纹理图像的宽度和高度
- 参数 border ：纹理边框尺寸
- 参数 format ：纹理映射格式，GLES20.GL_RGBA
- 参数 type ：数据格式，GL_UNSIGNED_BYTE
- 参数 pixels ：包含了纹理图像数据Buffer

纹理贴图

5. 激活纹理单元

GL_TEXTURE0默认激活，在使用其它纹理单元的时候需要手动激活。OpenGL ES支持的最小纹理单元与设备特性有关，通常情况下OpenGL ES2.0最少支持8个纹理单元。

```
glActiveTexture( GL_TEXTURE0 );
```

glActiveTextue 是选择当前活跃的纹理单元。

纹理贴图

6. 释放纹理对象

```
void glDeleteTextures(int n,
                      int[] texture,
                      int offset);
```

- 参数 n ：表示需要释放纹理对象的个数
- 参数 texture ：存储创建好的纹理对象句柄
- 参数 offset ：偏移量


THANK YOU