1. shader简介

这一章节首先对shader做一个简单的介绍，主要从以下几个方面：什么是shader？为什么要用shader？怎么用shader三方面来介绍。

这篇shader学习笔记来源于<https://thebookofshaders.com/?lan=ch>，可以直接读原文。

1. 什么是shader

shader是一段运行在GPU上的程序，它的出现主要是现代系统的发展，分辨率的提高，引入了GPU，而为了GPU的渲染，引入了shader，让其处理图形渲染相关的内容。

1. 为什么用shader

为了达到某种效果，我们需要直接操作GPU来达到我们的渲染效果，而shader是最方便的实现方式。

1. 怎么用shader

使用glsl语言来实现。

1. shader中的hello world

所有的语言学习都以hello world输出为基础来进行学习，而在shader中，直接输出一个文本是比较困难的操作，而指定整个区域为某一个固定的颜色反而是比较简单的，下面通过hello world来了解shader。

#ifdef GL\_ES

precision mediump float;

#endif

uniform float u\_time;

void main() {

gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0);

}

上述代码完成了将屏幕渲染成红色的操作。



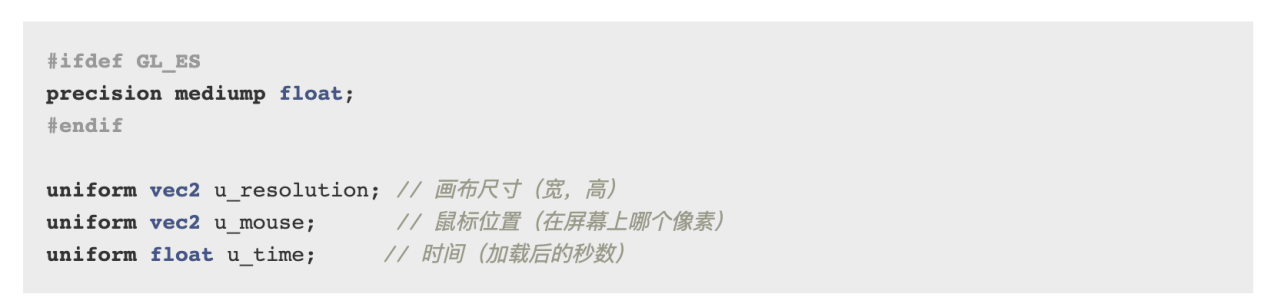
**以上是比较简单的shader渲染方式，而复杂一点的，我们需要通过操作空间(像素在屏幕中的位置)和时间 (页面加载的时长)。**

1. Uniform关键字

<https://thebookofshaders.com/03/?lan=ch>

在实际的操作中，线程与线程之间是无法交换数据的，但是他们统一接收GPU输入的数据。并且，由于显卡的架构问题，所有的输入值必须统一，且必须是只读的，所以通过uniform关键字来指明。

1. 定义方式

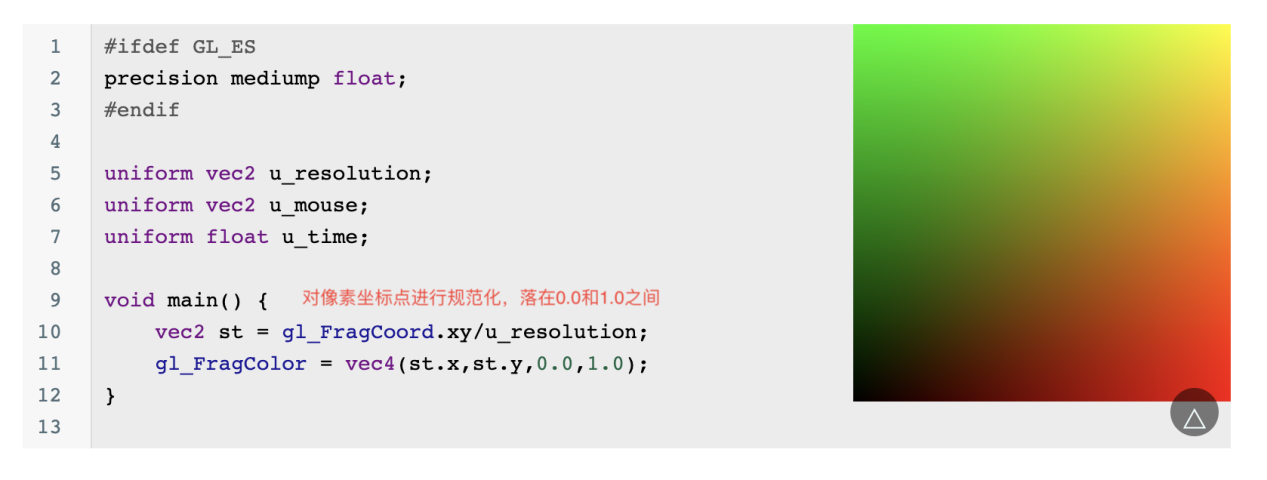


一般来说，习惯性的将uniform定义的变量名以u\_为前缀进行定义

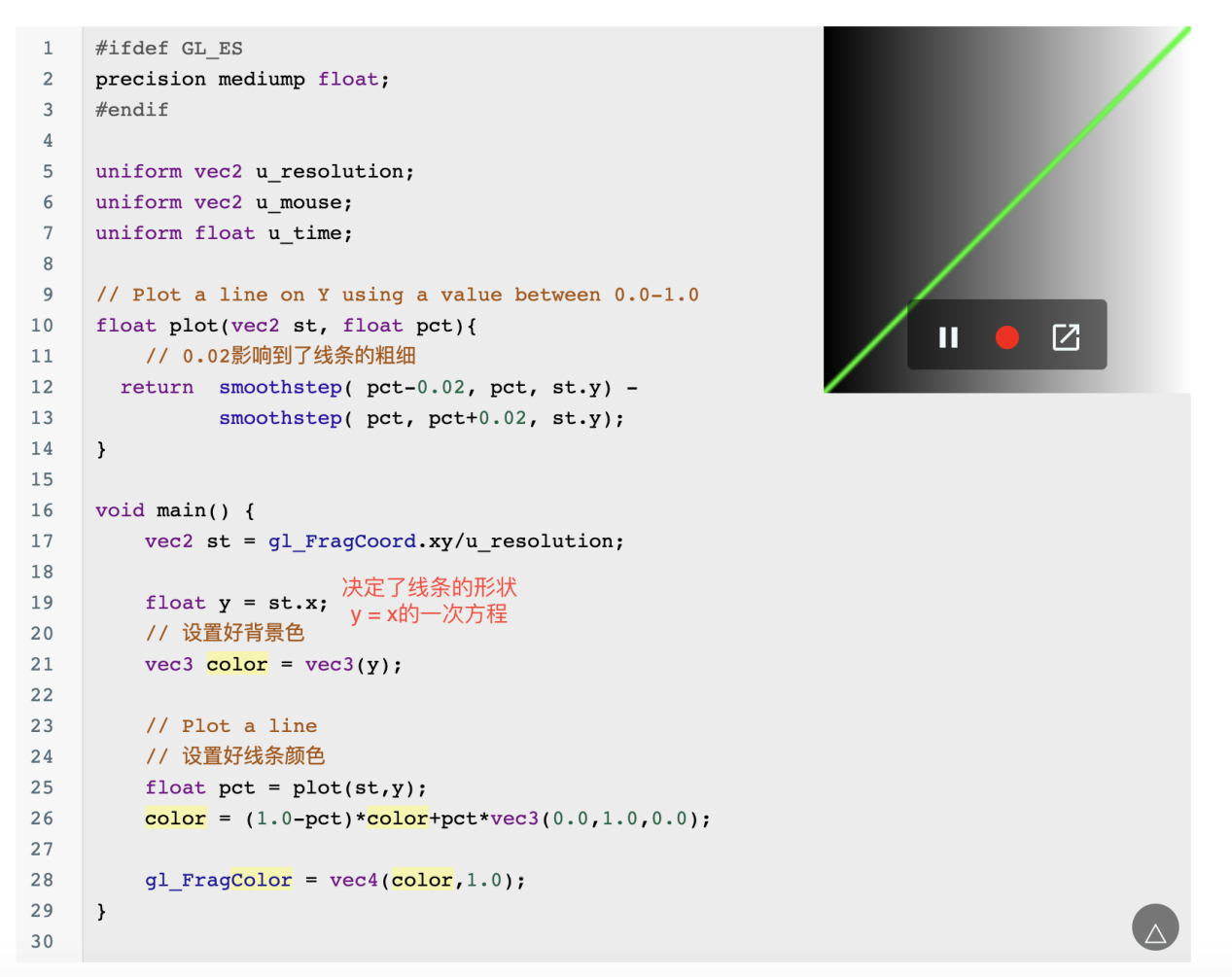


1. gl\_FragCoord关键字

该值存储了活动线程正在处理的像素或者屏幕碎片的坐标。



1. 用算法绘画
2. 造型函数



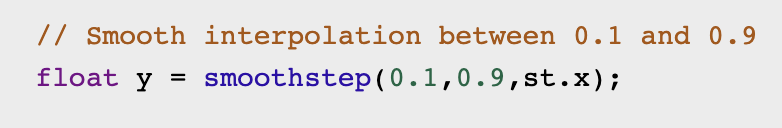
上述代码完成了右侧的效果。

1.1 Step和Smoothstep

这两个方法是线性插值方法。

Step：该方法返回两个值：0.0或者1.0，参数为阈值和我们要检测的值，将检测值<阈值的话，则返回0.0，当检测值>阈值的话，则返回1.0。

Smoothstep是在给定范围内给出插值。

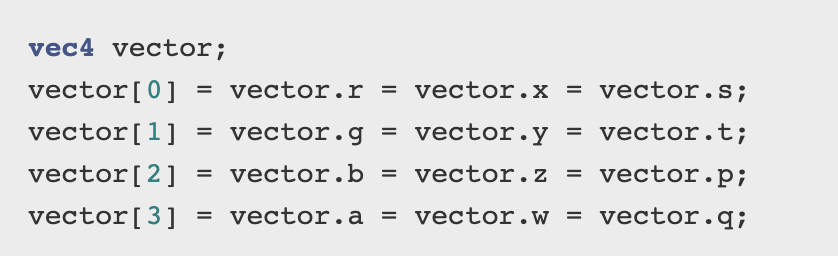


造型函数极大的帮助我们可以以一种优雅的方式控制像素的变化，不过要记住这么多东西基本不可能。慢慢来吧。

1. 颜色

首先我们需要通过某种形式来描述一个颜色，这里我们通过向量来描述即可。

glsl提供了4种方式来方便的访问颜色向量中的值。



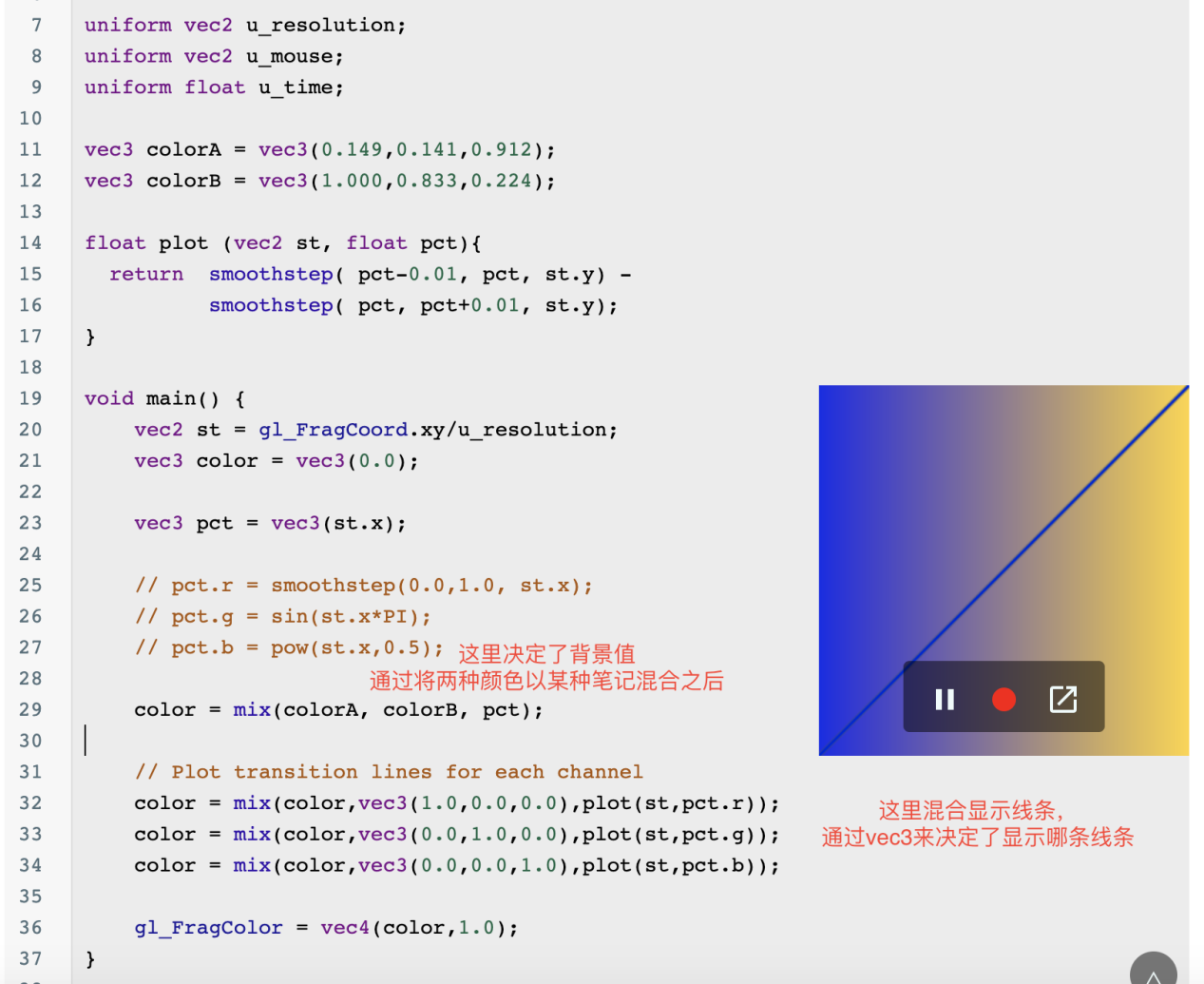
而向量的几个维度可以随意的组合，例如vec.rgb；vec.rb等等，都可以达到效果。

2.1 mix函数

mix函数可以满足以某个比例混合两个颜色的需要，基本用法



其中pct是混合的比例，让这个值一直变化，则可以实现颜色动态变化的效果。

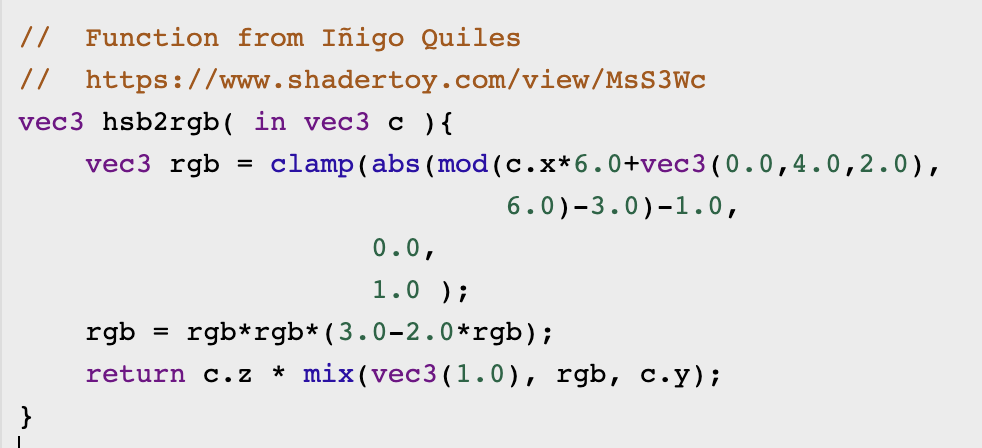


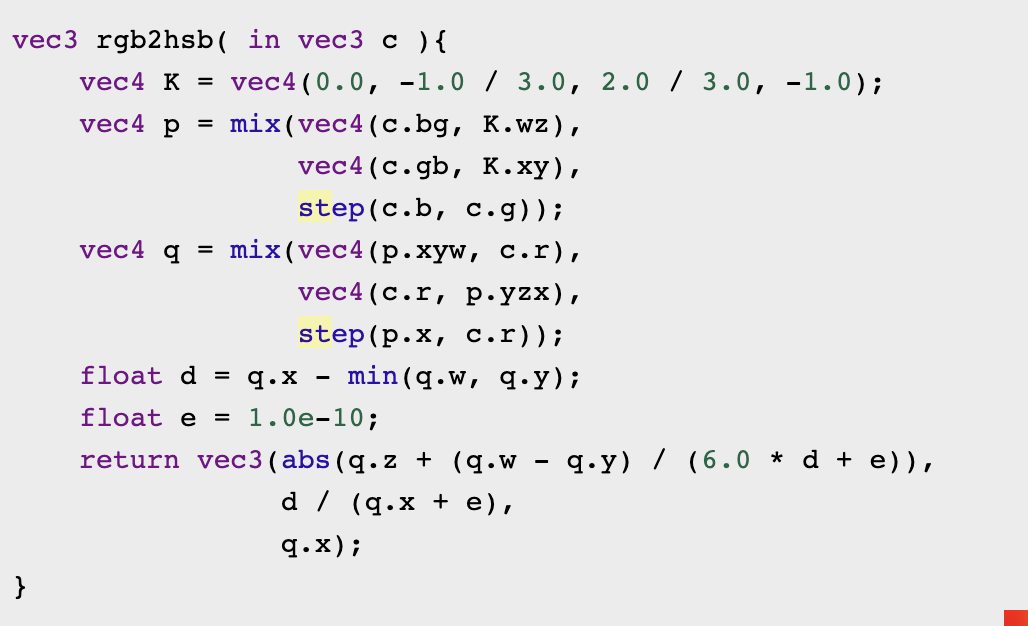
2.2 hsb

单纯的考虑颜色并不能真正的表达颜色，hsb考虑了颜色的更多的属性。

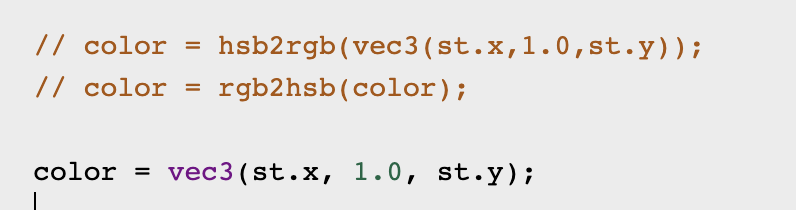
hsb代表了色相、饱和度和亮度。

将rgb值改为hsb值



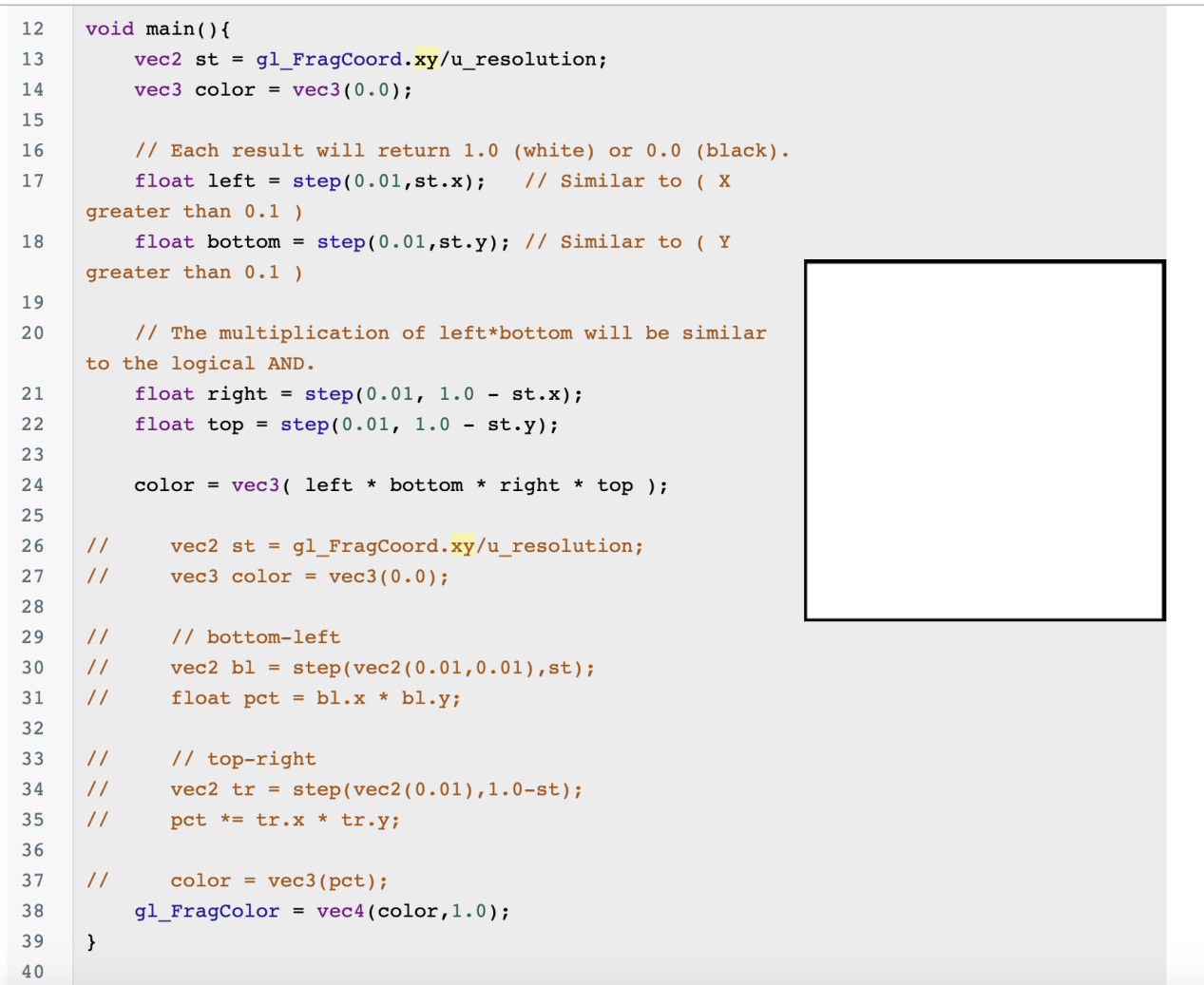


将hsb转换为rgb。



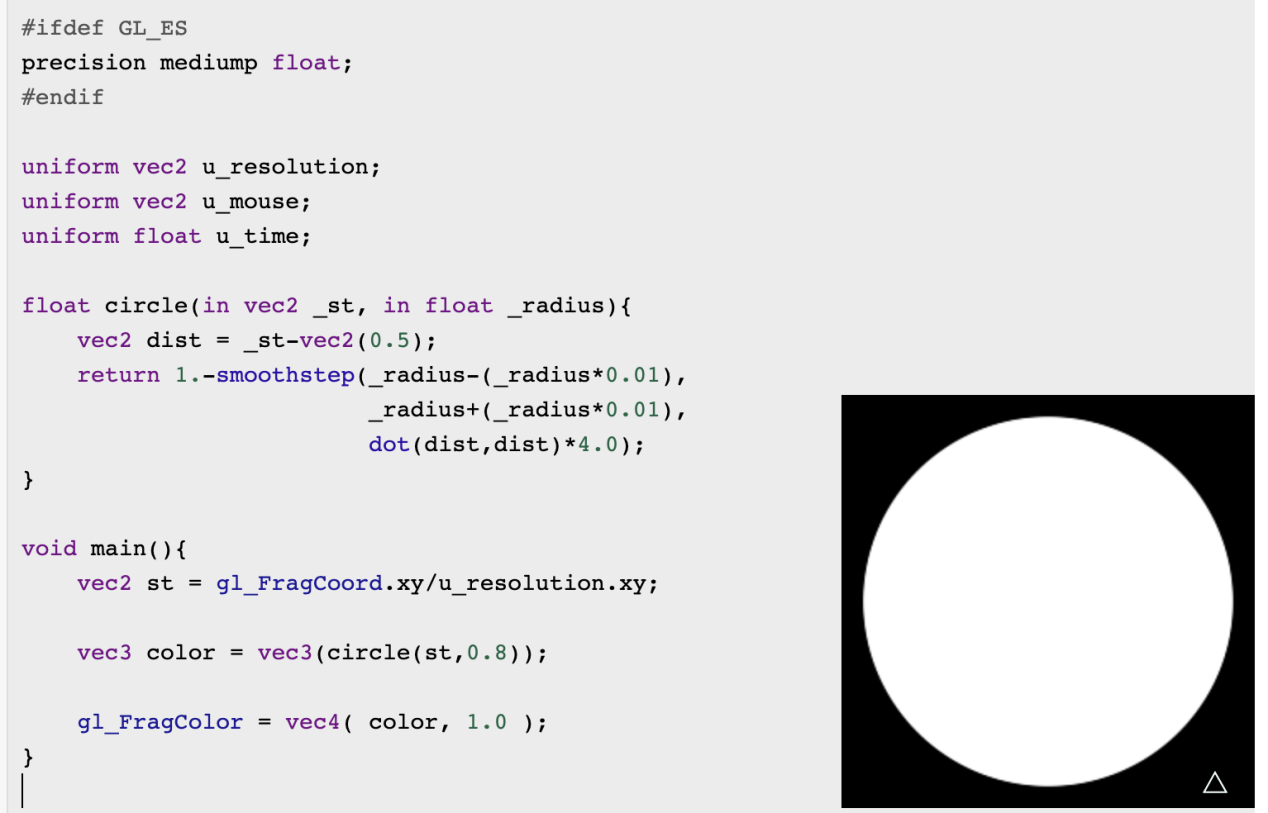
注意，上述部分，最后一行和上面两行实现的效果等价。

1. 形状



上述代码，上面的部分还下面的部分都可以实现画正方形。

注意正方形：左、右、上、下四个方面的使用。然后将color 再乘以一个vec3，就可以实现颜色填充。正方形变成不同的形状。



画一个圆形，当然，用最简单的step来画圆也可以，不过很明显锯齿特别严重。

1. 矩阵

我们需要对一个现有对图案进行平移、旋转、以及缩放。下面进行一一对应。这里矩阵的知识可以看下**云笔记中的opengl/数学知识篇**。

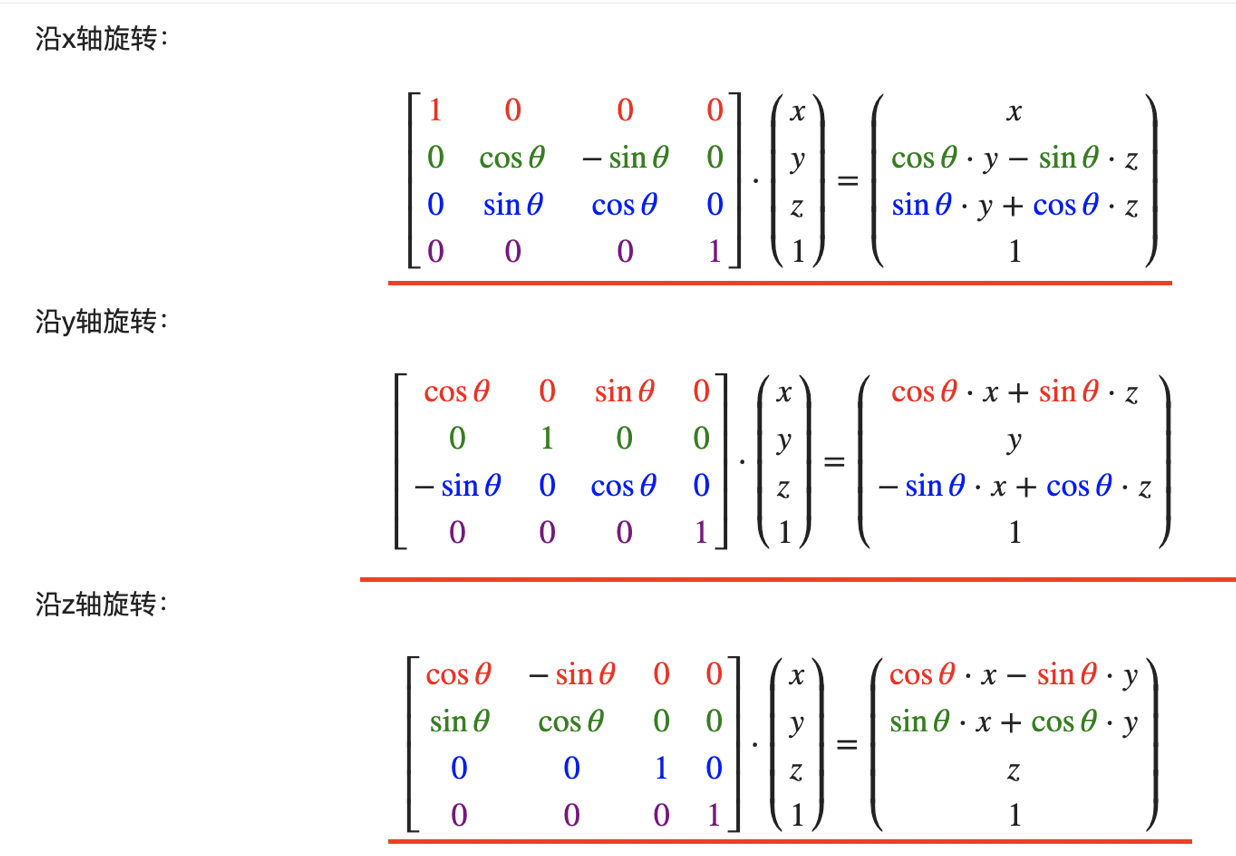
4.1 平移

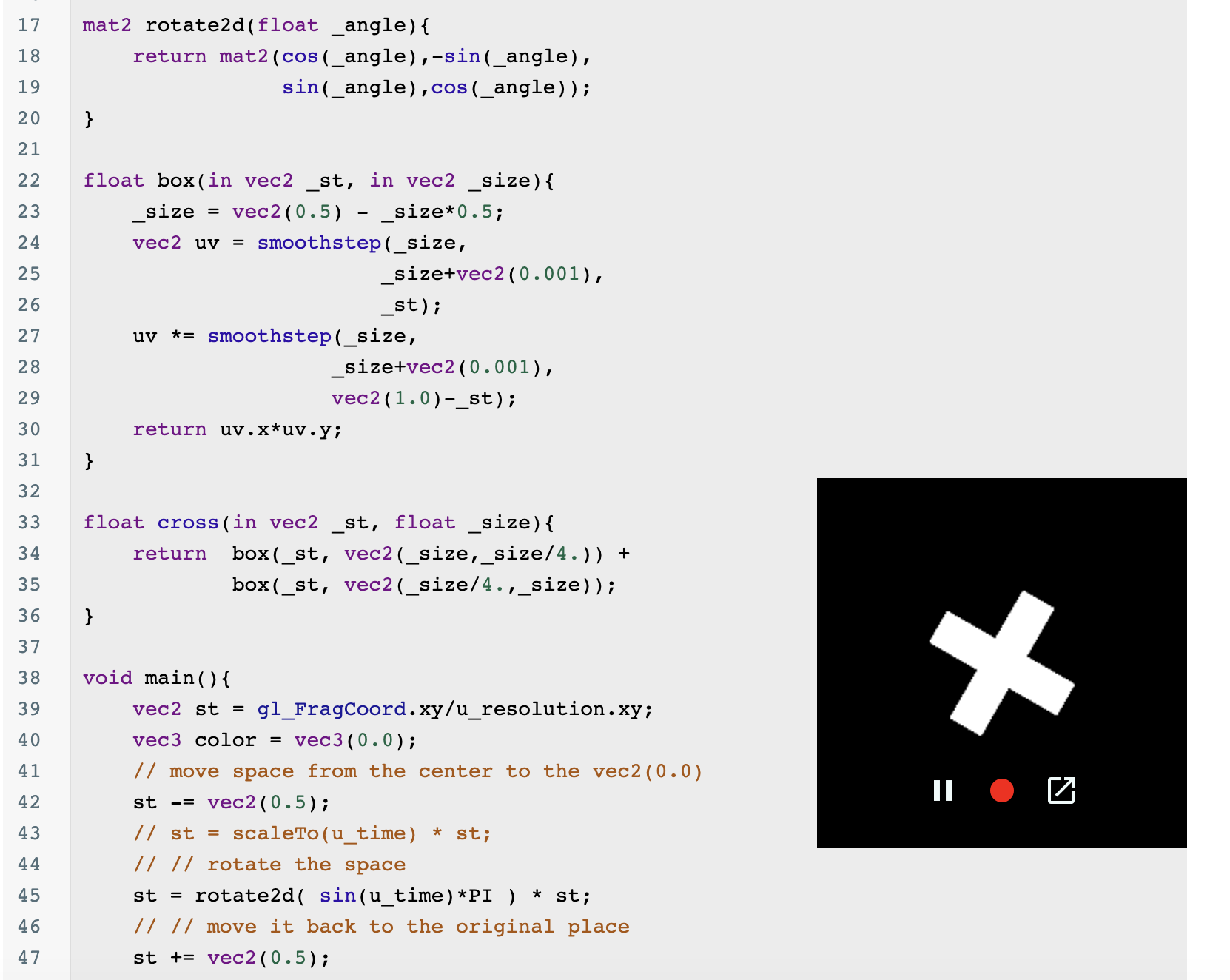
平移的话，简单点理解我们可以把自身的参考系进行偏移，则仅需要将我们计算的值进行一个偏移即可。



4.2 旋转

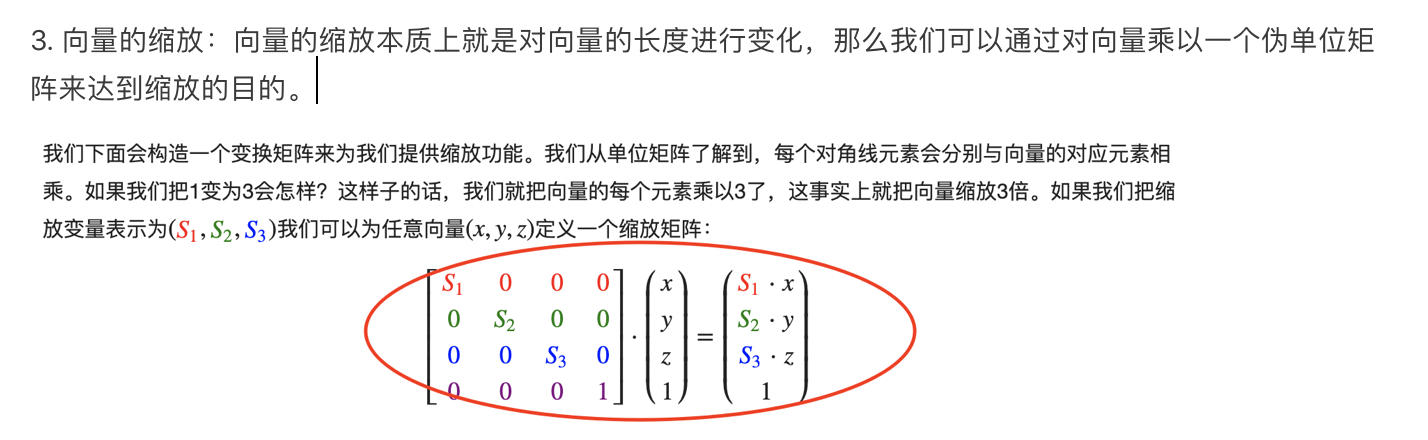
旋转的依据也是矩阵，依据不同的轴进行旋转，当然，也有一种万向节死锁。





4.3 缩放

缩放的话需要配合矩阵来达到效果，简单点来说，就是对单位矩阵进行变形操作。





通过上述的操作，可以实现形状原地旋转，先-后+。

1. 图案