# Creator3D:基础2\_一起学shader\_变色小球

# 前言

在上一篇《Creator3D:基础1\_一起学shader\_红色小球 》中简单讲了怎样在effect中给小球上色，有了这个基础，咱们是不是可以考虑让他的颜色变化起来

# 效果展示

![](https://gitee.com/carlosyzy/Creator3D\_Mesh\_Basics/raw/master/file/base2\_1.gif)

# 正文

## 1.文档资料

[YAML 101](https://docs.cocos.com/creator3d/manual/zh/material-system/yaml-101.html)

[Effect 语法](https://docs.cocos.com/creator3d/manual/zh/material-system/effect-syntax.html)

[Pass 可选配置参数](https://docs.cocos.com/creator3d/manual/zh/material-system/pass-parameter-list.html)

[常用 shader 内置 Uniform](https://docs.cocos.com/creator3d/manual/zh/material-system/builtin-shader-uniforms.html)

文档是必须要读的，哪怕你是和我一下的小白，第一次不要求看懂，但是最好过一遍，有个印象，方便后边回过头来查找

## 2.变色（cc\_time）

要让颜色动起来，那么需要做的就是实时的去设置颜色值

在这里给大家介绍第一个内置变量：cc\_time

![](https://gitee.com/carlosyzy/Creator3D\_Mesh\_Basics/raw/master/file/base2\_2.png)

游戏的运行时间，也就是shader的运行时间

我这边让颜色的r值按照运行时间的正弦函数值的绝对值变化：

```

vec4 frag () {

vec4 oldColor=vec4(abs(sin(cc\_time.x)),0.2,0.2,1.0);

return oldColor;

}

```

## 3.Fragment Ouput

在之前的文章和今天这个文章的我对颜色的返回都是直接return：

```

return oldColor;

```

当这样是不标准的，来看看官方的文档（[Effect语法](https://github.com/cocos-creator/docs-3d/blob/master/zh/material-system/effect-syntax.md)）

![](https://gitee.com/carlosyzy/Creator3D\_Mesh\_Basics/raw/master/file/base2\_3.png)

```

return CCFragOutput(oldColor);

```

需要注意的是试用CCFragOutput需要引入头文件最终的代码：

```

CCEffect %{

techniques:

- name: opaque

passes:

- vert: general-vs:vert # builtin header

frag: unlit-fs:frag

}%

CCProgram unlit-fs %{

precision highp float;

#include <cc-global>

#include <output>

vec4 frag () {

vec4 oldColor=vec4(abs(sin(cc\_time.x)),0.2,0.2,1.0);

return CCFragOutput(oldColor);

}

}%

```

## 4.内置函数

上边用到了sin函数，这里顺带给大家提供一下shader的内置函数

```

radians(degree) : 角度变弧度；

degrees(radian) : 弧度变角度；

sin(angle), cos(angle), tan(angle)

asin(x): arc sine, 返回弧度 [-PI/2, PI/2];

acos(x): arc cosine,返回弧度 [0, PI];

atan(y, x): arc tangent, 返回弧度 [-PI, PI];

atan(y/x): arc tangent, 返回弧度 [-PI/2, PI/2];

pow(x, y): x的y次方；

exp(x): 指数, log(x)：

exp2(x): 2的x次方， log2(x):

sqrt(x): x的根号； inversesqrt(x): x根号的倒数

abs(x): 绝对值

sign(x): 符号, 1, 0 或 -1

floor(x): 底部取整

ceil(x): 顶部取整

fract(x): 取小数部分

mod(x, y): 取模， x - y\*floor(x/y)

min(x, y): 取最小值

max(x, y): 取最大值

clamp(x, min, max): min(max(x, min), max);

mix(x, y, a): x, y的线性混叠， x(1-a) + y\*a;

step(edge, x): 如 x

smoothstep(edge0, edge1, x): threshod smooth transition时使用。 edge0<=edge0时为0.0， x>=edge1时为1.0

length(x): 向量长度

distance(p0, p1): 两点距离， length(p0-p1);

dot(x, y): 点积，各分量分别相乘 后 相加

cross(x, y): 差积，x[1]\*y[2]-y[1]\*x[2], x[2]\*y[0] - y[2]\*x[0], x[0]\*y[1] - y[0]\*x[1]

normalize(x): 归一化， length(x)=1;

faceforward(N, I, Nref): 如 dot(Nref, I)< 0则N, 否则 -N

reflect(I, N): I的反射方向， I -2\*dot(N, I)\*N, N必须先归一化

refract(I, N, eta): 折射，k=1.0-eta\*eta\*(1.0 - dot(N, I) \* dot(N, I)); 如k<0.0 则0.0，否则 eta\*I - (eta\*dot(N, I)+sqrt(k))\*N

matrixCompMult(matX, matY): 矩阵相乘, 每个分量 自行相乘， 即 r[i][j] = x[i][j]\*y[i][j];矩阵线性相乘，直接用 \*

lessThan(vecX, vecY): 向量 每个分量比较 x < y

lessThanEqual(vecX, vecY): 向量 每个分量比较 x<=y

greaterThan(vecX, vecY): 向量 每个分量比较 x>y

greaterThanEqual(vecX, vecY): 向量 每个分量比较 x>=y

equal(vecX, vecY): 向量 每个分量比较 x==y

notEqual(vecX, vexY): 向量 每个分量比较 x!=y

any(bvecX): 只要有一个分量是true， 则true

all(bvecX): 所有分量是true， 则true

not(bvecX): 所有分量取反

texture2D(sampler2D, coord): texture lookup

texture2D(sampler2D, coord, bias): LOD bias, mip-mapped texture

texture2DProj(sampler2D, coord):

texture2DProj(sampler2D, coord, bias):

texture2DLod(sampler2D, coord, lod):

texture2DProjLod(sampler2D, coord, lod):

textureCube(samplerCube, coord):

textureCube(samplerCube, coord, bias):

textureCubeLod(samplerCube, coord, lod):

```

具体的本人没有按个去测试，在后边的使用中用到了在验证，大家可以先过目一下

## 5.地址

- gitee：https://gitee.com/carlosyzy/Creator3D\_Mesh\_Basics

- 微信公众号：搬砖小菜鸟

![](https://gitee.com/carlosyzy/cocos\_creator\_physical\_cutting/raw/master/file/4.png)

## 6.推荐

推荐一款比较好玩的游戏

![](https://gitee.com/carlosyzy/Creator3D\_Mesh\_Basics/raw/master/file/fruit.jpg)