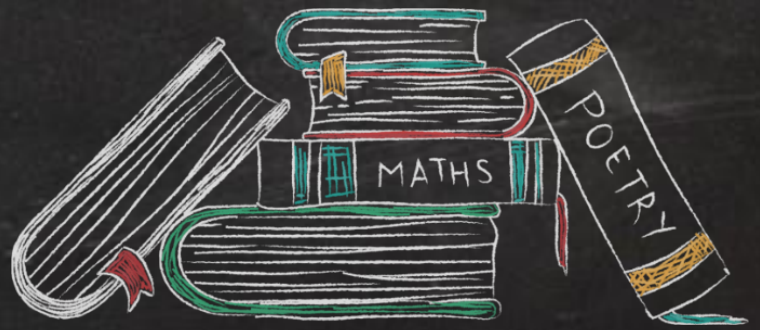
今日内容

- 符文·筑基·顶点平移
- 符文·筑基·顶点缩放
- 符文·筑基·顶点旋转
- 符文·化神·幽灵夜巡

$\frac{24 \times 5}{33}$

1 符文·筑基·顶点平移



01 案例

- AB基础上，追加Y轴向上周期性位移；

02 代码实现

1. 以L13_AB为模板，CtrlCV；
2. 修改路径名；
3. 追加面板参数：
    1. _MoveRange：移动范围；
    2. _MoveSpeed：移动速度；
4. SubShaderTags：不用改；
5. 混合模式：不用改；
6. 对应声明输入参数；

2

```
Shader "AP01/L19/Translation" {
    Properties {
        _MainTex        ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        _Opacity        ("透明度", range(0, 1)) = 0.5
        _MoveRange      ("移动范围", range(0.0, 3.0)) = 1.0
        _MoveSpeed      ("移动速度", range(0.0, 3.0)) = 1.0
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                   // 调整渲染顺序
            "RenderType"="Transparent"              // 对应改为Cutout
            "ForceNoShadowCasting"="True"           // 关闭阴影投射
            "IgnoreProjector"="True"                // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha              // 修改混合方式One/SrcAlpha OneMinusSrcAlpha

            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入参数
            uniform sampler2D _MainTex; uniform float4 _MainTex_ST;
            uniform half _Opacity;
            uniform float _MoveRange;
            uniform float _MoveSpeed;
```

3

6

03 代码实现

7. 输入结构，不用改；

8. 输出结构，不用改；

9. 声明常量：2π；

10. 声明顶点平移方法：void Translation(...)

11. 顶点Shader：追加对顶点位置信息的预处理：

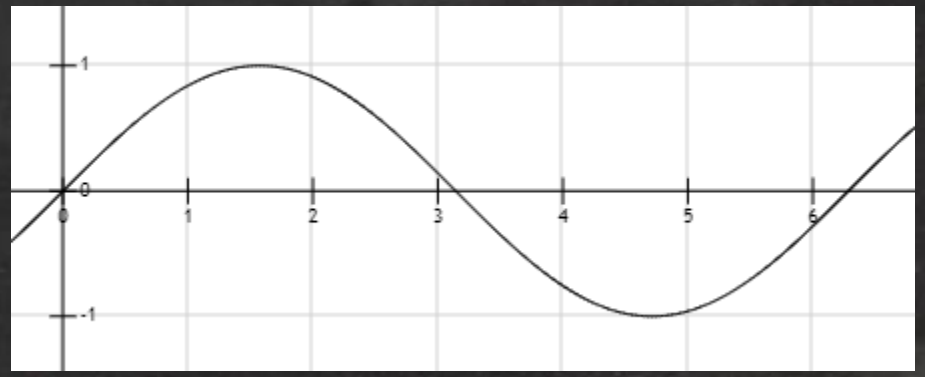
12. 像素Shader：不用改。

```
        // 输入结构
        struct VertexInput {
            float4 vertex : POSITION;        // 顶点位置 总是必要
            float2 uv : TEXCOORD0;           // UV信息 采样贴图用
        };
        // 输出结构
        struct VertexOutput {
            float4 pos : SV_POSITION;        // 顶点位置 总是必要
            float2 uv : TEXCOORD0;           // UV信息 采样贴图用
        };
        // 声明常量
9       #define TWO_PI 6.283185
        // 顶点动画方法
        void Translation (inout float3 vertex) {
10          vertex.y += _MoveRange * sin(frac(_Time.z * _MoveSpeed) * TWO_PI);
        }
        // 输入结构>>>顶点Shader>>>输出结构
        VertexOutput vert (VertexInput v) {
            VertexOutput o = (VertexOutput)0;
11          Translation(v.vertex.xyz);
            o.pos = UnityObjectToClipPos(v.vertex);     // 顶点位置 OS>CS
            o.uv = TRANSFORM_TEX(v.uv, _MainTex);       // UV信息 支持TilingOffset
            return o;
        }
        // 输出结构>>>像素
        half4 frag(VertexOutput i) : COLOR {
            half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
            half3 finalRGB = var_MainTex.rgb;
            half opacity = var_MainTex.a * _Opacity;
            return half4(finalRGB * opacity, opacity);                  // 返回值
        }
        ENDCG
    }
}
}
```

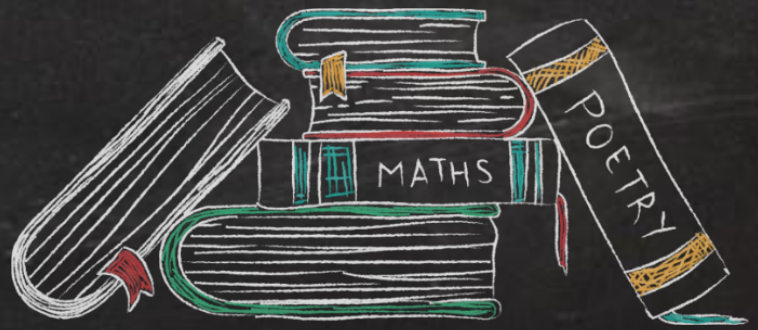
04 核心代码分析

```
// 声明常量
#define TWO_PI 6.283185
// 顶点动画方法
void Translation (inout float3 vertex) {
    vertex.y += _MoveRange * sin(frac(_Time.z * _MoveSpeed) * TWO_PI);
}
```

1. 常量声明方法：#define 常量名 常量值
2. void声明无返回值方法；
3. Inout修饰：参数的出入证；
4. frac(...)：浮点精度保护；
5. sin(...)：产生波动的常用方法；
6. 计算偏移值，并加到顶点对应轴，以实现平移效果；



2 符文·筑基·顶点缩放



01 案例

- AB基础上，基于模型原点追加周期性缩放；

02 代码实现

1. 以L13_AB为模板，CtrlCV；
2. 修改路径名；
3. 追加面板参数：
    1. _ScaleRange：移动范围；
    2. _ScaleSpeed：移动速度；
4. SubShaderTags：不用改；
5. 混合模式：不用改；
6. 对应声明输入参数；

2

```
Shader "AP01/L19/Scaling" {
    Properties {
        _MainTex        ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        _Opacity        ("透明度", range(0, 1)) = 0.5
        _ScaleRange     ("缩放范围", range(0.0, 3.0)) = 1.0
        _ScaleSpeed     ("缩放速度", range(0.0, 3.0)) = 1.0
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                // 调整渲染顺序
            "RenderType"="Transparent"           // 对应改为Cutout
            "ForceNoShadowCasting"="True"        // 关闭阴影投射
            "IgnoreProjector"="True"             // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha           // 修改混合方式One/SrcAlpha OneMinusSrcAlpha

            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入参数
            uniform sampler2D _MainTex; uniform float4 _MainTex_ST;
            uniform half _Opacity;
            uniform float _ScaleRange;
            uniform float _ScaleSpeed;
```

3

6

03 代码实现

7. 输入结构，不用改；
8. 输出结构，不用改；
9. 声明常量：2π；
10. 声明顶点缩放方法：void Scaling(...)
11. 顶点Shader：追加对顶点位置信息的预处理：
12. 像素Shader：不用改。

```
            // 输入结构
            struct VertexInput {
                float4 vertex : POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 输出结构
            struct VertexOutput {
                float4 pos : SV_POSITION;        // 顶点位置 总是必要
                float2 uv : TEXCOORD0;           // UV信息 采样贴图用
            };
            // 声明常量
            #define TWO_PI 6.283185
            // 顶点动画方法
            void Scaling (inout float3 vertex) {
                vertex *= 1.0 + _ScaleRange * sin(frac(_Time.z * _ScaleSpeed) * TWO_PI);
            }
            // 输入结构>>>顶点Shader>>>输出结构
            VertexOutput vert (VertexInput v) {
                VertexOutput o = (VertexOutput)0;
                    Scaling(v.vertex.xyz);
                    o.pos = UnityObjectToClipPos(v.vertex);     // 顶点位置 OS>CS
                    o.uv = TRANSFORM_TEX(v.uv, _MainTex);       // UV信息 支持TilingOffset
                return o;
            }
            // 输出结构>>>像素
            half4 frag(VertexOutput i) : COLOR {
                half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
                half3 finalRGB = var_MainTex.rgb;
                half opacity = var_MainTex.a * _Opacity;
                return half4(finalRGB * opacity, opacity);                  // 返回值
            }
            ENDCG
        }
    }
}
```

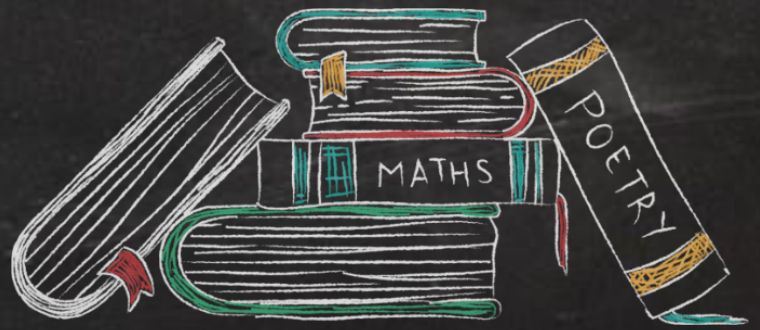
9
10
11

04 核心代码分析

```
// 顶点动画方法
void Scaling (inout float3 vertex) {
    vertex *= 1.0 + _ScaleRange * sin(frac(_Time.z * _ScaleSpeed) * TWO_PI);
}
```

1. 计算缩放比例，并乘到对应坐标轴，以实现缩放效果。

3 符文·筑基·顶点旋转



01 案例

- AB基础上，基于模型Y轴追加周期性旋转；

02 代码实现

1. 以L13_AB为模板，CtrlCV；
2. 修改路径名；
3. 追加面板参数：
    1. _RotateRange：移动范围；
    2. _RotateSpeed：移动速度；
4. SubShaderTags：不用改；
5. 混合模式：不用改；
6. 对应声明输入参数；

2

```
Shader "AP01/L19/Rotation" {
    Properties {
        _MainTex        ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        _Opacity        ("透明度", range(0, 1)) = 0.5
        _RotateRange    ("旋转范围", range(0.0, 45.0)) = 20.0
        _RotateSpeed    ("旋转速度", range(0.0, 3.0)) = 1.0
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                   // 调整渲染顺序
            "RenderType"="Transparent"              // 对应改为Cutout
            "ForceNoShadowCasting"="True"           // 关闭阴影投射
            "IgnoreProjector"="True"                // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha              // 修改混合方式One/SrcAlpha OneMinusSrcAlpha

            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入参数
            uniform sampler2D _MainTex; uniform float4 _MainTex_ST;
            uniform half _Opacity;
            uniform float _RotateRange;
            uniform float _RotateSpeed;
```

3

6

03 代码实现

7. 输入结构，不用改；

8. 输出结构，不用改；

9. 声明常量：2π；

10. 声明顶点旋转方法：void Rotation(...)

11. 顶点Shader：追加对顶点位置信息的预处理：

10. 像素Shader：不用改。

```
// 输入结构
struct VertexInput {
    float4 vertex : POSITION;        // 顶点位置 总是必要
    float2 uv : TEXCOORD0;           // UV信息 采样贴图用
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION;        // 顶点位置 总是必要
    float2 uv : TEXCOORD0;           // UV信息 采样贴图用
};
// 声明常量
#define TWO_PI 6.283185
// 顶点动画方法
void Rotation (inout float3 vertex) {
    float angleY = _RotateRange * sin(frac(_Time.z * _RotateSpeed) * TWO_PI);
    float radY = radians(angleY);
    float sinY, cosY = 0;
    sincos(radY, sinY, cosY);
    vertex.xz = float2(
        vertex.x * cosY - vertex.z * sinY,
        vertex.x * sinY + vertex.z * cosY
    );
}
// 输入结构>>>顶点Shader>>>输出结构
VertexOutput vert (VertexInput v) {
    VertexOutput o = (VertexOutput)0;
        Rotation(v.vertex.xyz);
        o.pos = UnityObjectToClipPos(v.vertex);    // 顶点位置 OS>CS
        o.uv = TRANSFORM_TEX(v.uv, _MainTex);      // UV信息 支持TilingOffset
    return o;
}
// 输出结构>>>像素
half4 frag(VertexOutput i) : COLOR {
    half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
    half3 finalRGB = var_MainTex.rgb;
    half opacity = var_MainTex.a * _Opacity;
    return half4(finalRGB * opacity, opacity);                  // 返回值
}
ENDCG
    }
  }
}
```

9

10

11

04 核心代码分析

```
// 声明常量
#define TWO_PI 6.283185
// 顶点动画方法
void Rotation (inout float3 vertex) {
    float angleY = _RotateRange * sin(frac(_Time.z * _RotateSpeed) * TWO_PI);
    float radY = radians(angleY);
    float sinY, cosY = 0;
    sincos(radY, sinY, cosY);
    vertex.xz = float2(
        vertex.x * cosY - vertex.z * sinY,
        vertex.x * sinY + vertex.z * cosY
    );
}
```

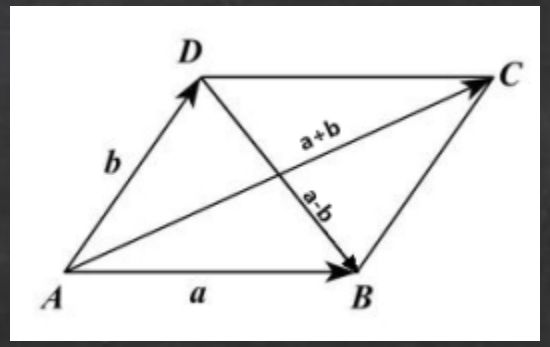
1

2

3

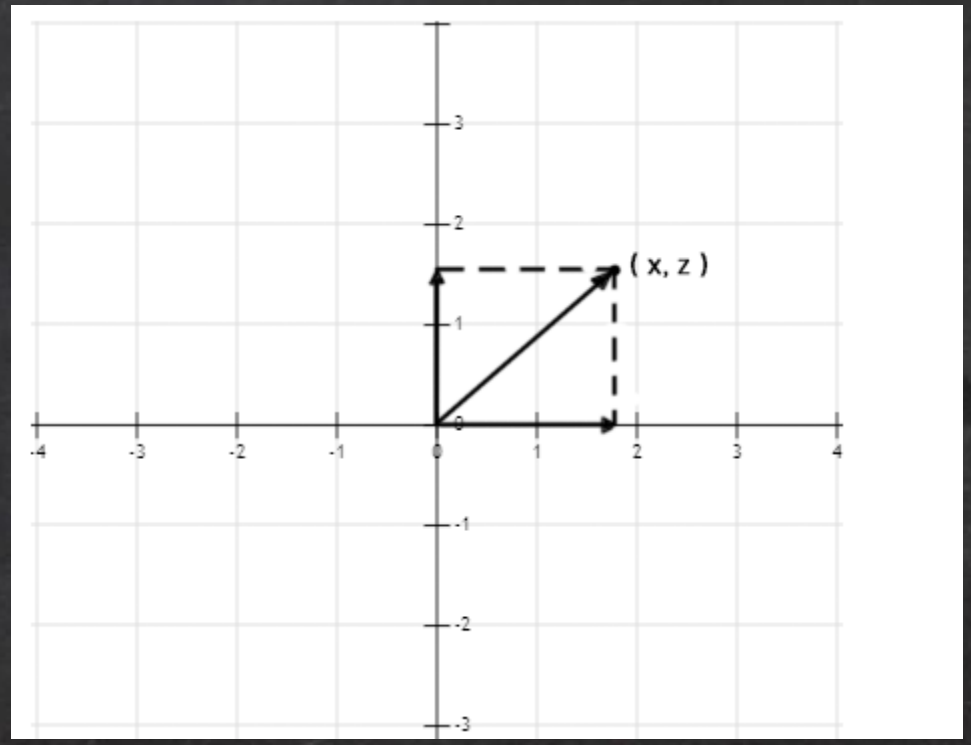
1. 计算偏转角度;
2. 角度转弧度 (使用三角函数时，注意参数为弧度)，计算Sin Cos值;
3. 换算偏转后的X Z轴值 (沿Y轴选择，估Y轴值不变);

05 向量加减法

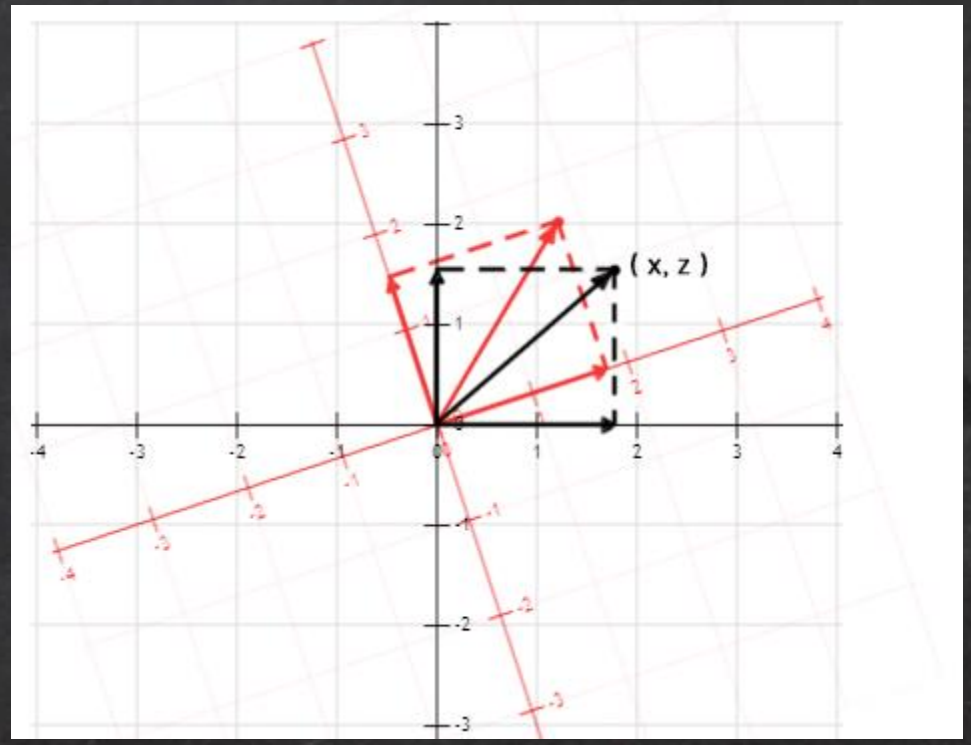


- 加法：将两个向量平移至公共起点，以向量的两条边作平行四边形，结果为公共起点的对角线；
- 减法：将两个向量平移至公共起点，以向量的两条边作平行四边形，结果由减向量的终点指向被减向量的终点。

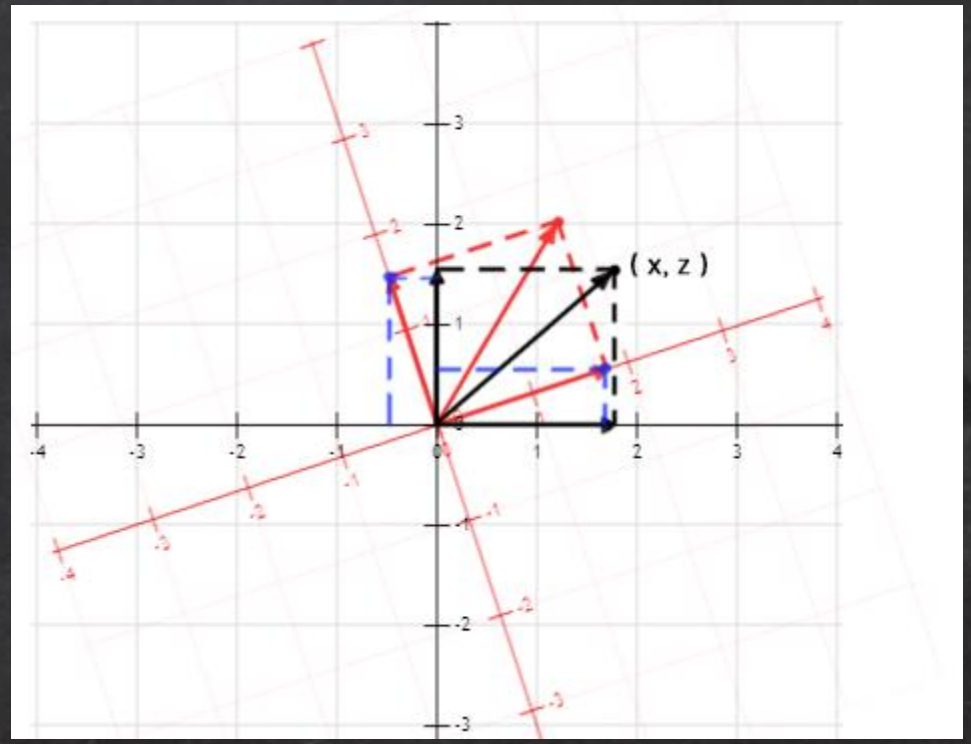
06 二维向量沿原点旋转



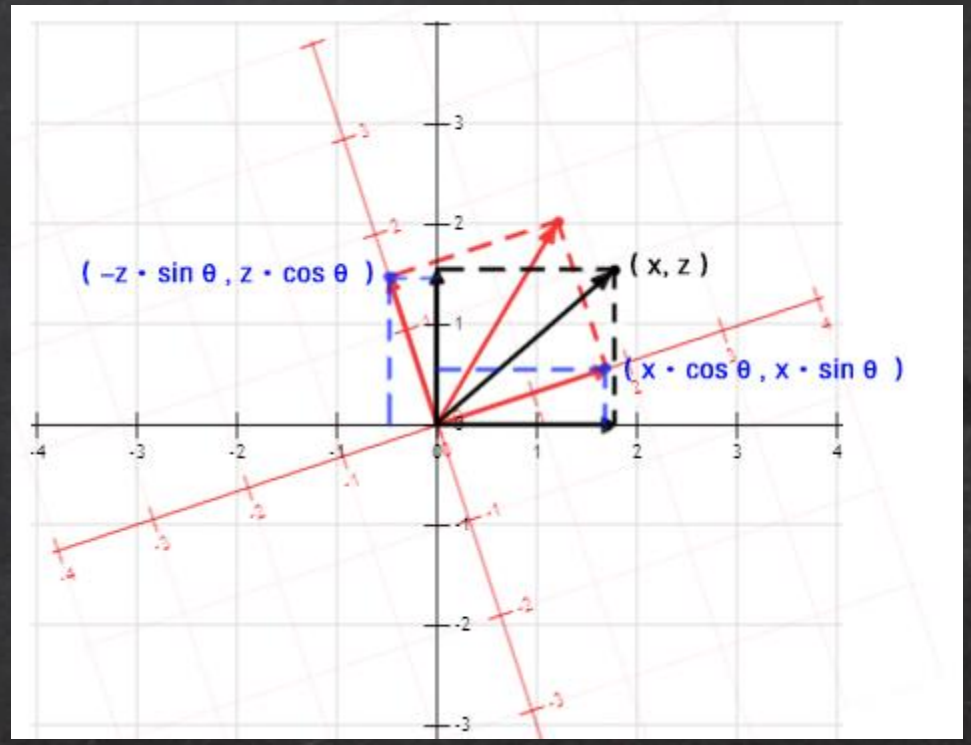
06 二维向量沿原点旋转



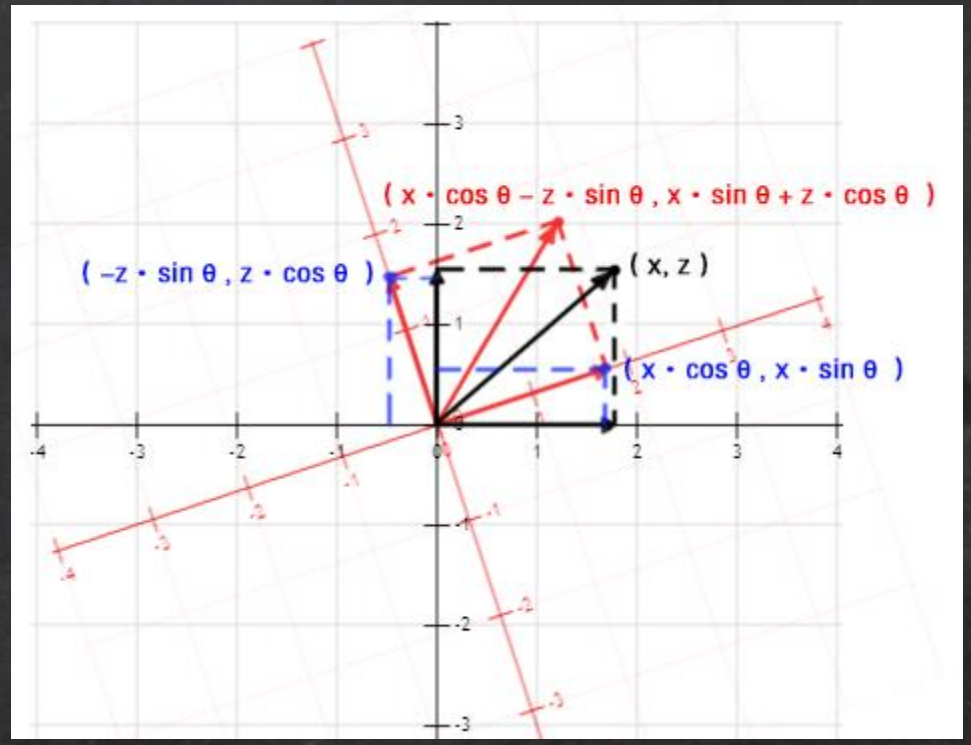
06 二维向量沿原点旋转



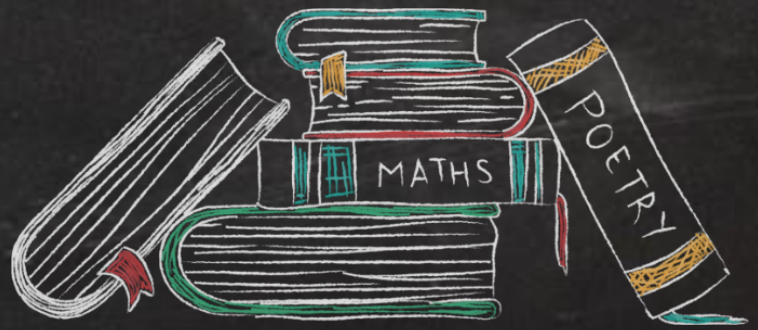
06 二维向量沿原点旋转



06 二维向量沿原点旋转



4 符文·化神·幽灵夜巡



01 案例

- AB基础上，混合各种基本动画形式为复杂动画；
- 动画拆分：
  1. 天使圈：缩放；
  2. 身子：X轴摆动，Z轴摆动；
  3. 头部：Y轴旋转；
  4. 全身：Y轴起伏。

02 物料准备

- 如图绘制模型顶点色：
  - R：身体摆动遮罩；
  - G：天使圈遮罩；

03 代码实现

1. 以L13_AB为模板，CtrlCV；
2. 修改路径名；
3. 追加面板参数：
    1. _ScaleParams：天使圈缩放相关参数；
    2. _SwingXParams：X轴扭动相关参数；
    3. _SwingZParams：Z轴扭动相关参数；
    4. _SwingYParams：Y轴起伏相关参数；
    5. _ShakeYParams：Y轴摇头相关参数；
4. SubShaderTags：不用改；
5. 混合模式：不用改；
6. 对应声明输入参数；

2

```
Shader "AP01/L19/AnimGhost" {
    Properties {
        _MainTex        ("RGB: 颜色 A: 透贴", 2d) = "gray"{}
        _Opacity        ("透明度", range(0, 1)) = 0.5
        _ScaleParams    ("天使圈缩放 X:强度 Y:速度 Z:校正", vector) = (0.2, 1.0, 4.5, 0.0)
        _SwingXParams   ("X轴扭动 X:强度 Y:速度 Z:波长", vector) = (1.0, 3.0, 1.0, 0.0)
        _SwingZParams   ("Z轴扭动 X:强度 Y:速度 Z:波长", vector) = (1.0, 3.0, 1.0, 0.0)
        _SwingYParams   ("Y轴起伏 X:强度 Y:速度 Z:滞后", vector) = (1.0, 3.0, 0.3, 0.0)
        _ShakeYParams   ("Y轴摇头 X:强度 Y:速度 Z:滞后", vector) = (20.0, 3.0, 0.3, 0.0)
    }
    SubShader {
        Tags {
            "Queue"="Transparent"                   // 调整渲染顺序
            "RenderType"="Transparent"              // 对应改为Cutout
            "ForceNoShadowCasting"="True"           // 关闭阴影投射
            "IgnoreProjector"="True"                // 不响应投射器
        }
        Pass {
            Name "FORWARD"
            Tags {
                "LightMode"="ForwardBase"
            }
            Blend One OneMinusSrcAlpha              // 修改混合方式One/SrcAlpha OneMinusSrcAlpha

            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #include "UnityCG.cginc"
            #pragma multi_compile_fwdbase_fullshadows
            #pragma target 3.0
            // 输入参数
            uniform sampler2D _MainTex; uniform float4 _MainTex_ST;
            uniform half _Opacity;
            uniform float4 _ScaleParams;
            uniform float3 _SwingXParams;
            uniform float3 _SwingZParams;
            uniform float3 _SwingYParams;
            uniform float3 _ShakeYParams;
```

3

6

04 代码实现

7. 输入结构，追加顶点色；
8. 输出结构，追加顶点色；
9. 声明常量：2π；
10. 声明动画方法：void AnimGhost(...)
    1. 天使圈缩放；
    2. 幽灵摆动；
    3. 幽灵摇头；
    4. 幽灵起伏；
    5. 处理顶点色；

7
8
9
10

```
// 输入结构
struct VertexInput {
    float4 vertex : POSITION;       // 顶点位置 总是必要
    float2 uv : TEXCOORD0;          // UV信息 采样贴图用
    float4 color : COLOR;           // 顶点色 遮罩用
};
// 输出结构
struct VertexOutput {
    float4 pos : SV_POSITION;       // 顶点位置 总是必要
    float2 uv : TEXCOORD0;          // UV信息 采样贴图用
    float4 color : COLOR;
};
// 声明常量
#define TWO_PI 6.283185
// 顶点动画方法
void AnimGhost (inout float3 vertex, inout float3 color) {
    // 缩放天使圈
    float scale = _ScaleParams.x * color.g * sin(frac(_Time.z * _ScaleParams.y) * TWO_PI);
    vertex.xyz *= 1.0 + scale;
    vertex.y -=  _ScaleParams.z * scale;
    // 幽灵摆动
    float swingX = _SwingXParams.x * sin(frac(_Time.z * _SwingXParams.y + vertex.y * _SwingXParams.z) * TWO_PI);
    float swingZ = _SwingZParams.x * sin(frac(_Time.z * _SwingZParams.y + vertex.y * _SwingZParams.z) * TWO_PI);
    vertex.xz += float2(swingX, swingZ) * color.r;
    // 幽灵摇头
    float radY = radians(_ShakeYParams.x) * (1.0 - color.r)
                * sin(frac(_Time.z * _ShakeYParams.y - color.g * _ShakeYParams.z) * TWO_PI);
    float sinY, cosY = 0;
    sincos(radY, sinY, cosY);
    vertex.xz = float2(
        vertex.x * cosY - vertex.z * sinY,
        vertex.x * sinY + vertex.z * cosY
    );
    // 幽灵起伏
    float swingY = _SwingYParams.x * sin(frac(_Time.z * _SwingYParams.y - color.g * _SwingYParams.z) * TWO_PI);
    vertex.y += swingY;
    // 处理顶点色
    float lightness = 1.0 + color.g * 1.0 + scale * 2.0;
    color = float3(lightness, lightness, lightness);
}
```

05 代码实现

11. 顶点Shader：追加对顶点位置信息的预处理；
10. 像素Shader：用顶点色实现天使圈亮度动画。

```
        // 输入结构>>>顶点Shader>>>输出结构
        VertexOutput vert (VertexInput v) {
            VertexOutput o = (VertexOutput)0;
                AnimGhost(v.vertex.xyz, v.color.rgb);
                o.pos = UnityObjectToClipPos(v.vertex);     // 顶点位置 OS>CS
                o.uv = TRANSFORM_TEX(v.uv, _MainTex);       // UV信息 支持TilingOffset
                o.color = v.color;
            return o;
        }
        // 输出结构>>>像素
        half4 frag(VertexOutput i) : COLOR {
            half4 var_MainTex = tex2D(_MainTex, i.uv);      // 采样贴图 RGB颜色 A透贴
            half3 finalRGB = var_MainTex.rgb * i.color.rgb;
            half opacity = var_MainTex.a * _Opacity;
            return half4(finalRGB * opacity, opacity);                  // 返回值
        }
        ENDCG
    }
}
}
```

11

12

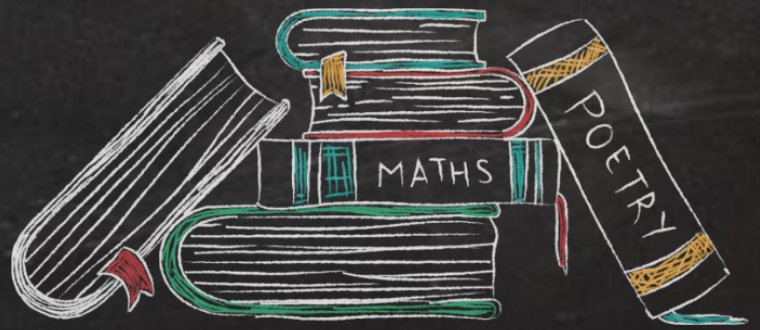
06 核心代码分析

```
// 顶点动画方法
void AnimGhost (inout float3 vertex, inout float3 color) {
    // 缩放天使圈
    float scale = _ScaleParams.x * color.g * sin(frac(_Time.z * _ScaleParams.y) * TWO_PI);
    vertex.xyz *= 1.0 + scale;
    vertex.y -=  _ScaleParams.z * scale;
    // 幽灵摆动
    float swingX = _SwingXParams.x * sin(frac(_Time.z * _SwingXParams.y + vertex.y * _SwingXParams.z) * TWO_PI);
    float swingZ = _SwingZParams.x * sin(frac(_Time.z * _SwingZParams.y + vertex.y * _SwingZParams.z) * TWO_PI);
    vertex.xz += float2(swingX, swingZ) * color.r;
    // 幽灵摇头
    float radY = radians(_ShakeYParams.x) * (1.0 - color.r) * sin(frac(_Time.z * _ShakeYParams.y - color.g * _ShakeYParams.z) * TWO_PI);
    float sinY, cosY = 0;
    sincos(radY, sinY, cosY);
    vertex.xz = float2(
        vertex.x * cosY - vertex.z * sinY,
        vertex.x * sinY + vertex.z * cosY
    );
    // 幽灵起伏
    float swingY = _SwingYParams.x * sin(frac(_Time.z * _SwingYParams.y - color.g * _SwingYParams.z) * TWO_PI);
    vertex.y += swingY;
    // 处理顶点色
    float lightness = 1.0 + color.g * 1.0 + scale * 2.0;
    color = float3(lightness, lightness, lightness);
}
```

1. 天使圈：用顶点色遮罩缩放动画；校正放缩后的位置；
2. 摆动：用Y轴值偏移出正弦波摆动；用顶点色遮罩摆动；
3. 摇头：用顶点色遮罩头部实现摇头；用顶点色实现天使圈滞后；
4. 起伏：用顶点色实现天使圈滞后；
5. 顶点色：计算天使圈亮度值；

$\frac{24 \times 5}{33}$

5 任务委托



01 作业

必做作业：

- 自己找模型实现 AnimGhost 效果；

创意作业：

- 课内知识自由发挥；