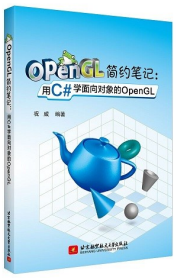
摩不有初，鲜克有终

» CSharpGL @ GitHub

» 我的博客汇总

» 捐赠列表

加入QQ群



昵称： BIT祝威
园龄： 7年5个月
荣誉： 推荐博客
粉丝： 765
关注： 12
+加关注

最新随笔

1.[译]可见性判断之门系统

2.《资本论》核心思想

3.[译]为任意网格计算tangent空间的基向量

4.[译]Vulkan教程(33)多重采样

5.[译]Vulkan教程(32)生成mipmap

6.[译]Vulkan教程(31)加载模型

7.[译]Vulkan教程(30)深度缓存

8.[译]Vulkan教程(29)组合的Image采样器

9.[译]Vulkan教程(28)Image视图和采样器

10.[译]Vulkan教程(27)Image

最新评论

1. Re:[译]Vulkan教程(33)多重采样

@ 幕三少引用 只支持c++吗? Vulkan和OpenGL一样，不限制语言。...

--BIT祝威

Opengl中矩阵和perspective/ortho的相互转换

阅读目录(Content)

Opengl中矩阵和perspective/ortho的相互转换

回到顶部(go to top)

定义矩阵

Opengl变换需要用四维矩阵。我们来定义这样的矩阵。

四维向量

首先，我们定义一个四维向量vec4。

```
/// <summary>
/// Represents a four dimensional vector.
/// </summary>
public struct vec4
{
    public float x;
    public float y;
    public float z;
    public float w;

    public float this[int index]
    {
        get
        {
            if (index == 0) return x;
            else if (index == 1) return y;
            else if (index == 2) return z;
            else if (index == 3) return w;
            else throw new Exception("Out of range.");
```

2. Re:CSharpGL(28)得到高精度可定制字形贴图的极简方法

@ 夜丶止不住的黑引用 楼主大大怎么识别中文字符 有没有案例啊 求指点 这不是识别文字的库。...

--BIT祝威

3. Re:CSharpGL(28)得到高精度可定制字形贴图的极简方法

@ 武胜-阿伟引用 文中提到：便于debug和观看效果，我在CSharpGL.Demos里加了下面这个Demo。你可以指定任意字体，设置是否启用加粗、斜体、下划线、删除线等效果。请问这个Demo的名称...

--BIT祝威

4. Re:CSharpGL(28)得到高精度可定制字形贴图的极简方法

文中提到：便于debug和观看效果，我在CSharpGL.Demos里加了下面这个Demo。你可以指定任意字体，设置是否启用加粗、斜体、下划线、删除线等效果。请问这个Demo的名称，在哪儿下载？谢谢，...

--武胜-阿伟

5. Re:[译]Vulkan教程(33)多重采样

只支持c++吗？

--幕三少

阅读排行榜

1. C#自定义控件：WinForm将其它应用程序窗体嵌入自己内部(36776)

2. 继电器是如何成为CPU的（1）(30379)

3. 图文详解Unity3D中Material的Tiling和Offset是怎么回事(18139)

4. 《30天自制操作系统》笔记(01)——hello bitzhuwei's OS!(16430)

5. CSharpGL(0)一个易学易用的C#版OpenGL(14957)

6. Unity3D核心类型一览(14176)

7. 《30天自制操作系统》笔记(02)——导入C语言(12860)

8. [译+改]最长回文子串(Longest Palindromic Substring) Part II(11234)

9. 《30天自制操作系统》笔记(03)——使用Vmware(10258)

```
20              }
21              set
22              {
23                  if (index == 0) x = value;
24                  else if (index == 1) y = value;
25                  else if (index == 2) z = value;
26                  else if (index == 3) w = value;
27                  else throw new Exception("Out of range.");
28              }
29          }
30
31          public vec4(float s)
32          {
33              x = y = z = w = s;
34          }
35
36          public vec4(float x, float y, float z, float w)
37          {
38              this.x = x;
39              this.y = y;
40              this.z = z;
41              this.w = w;
42          }
43
44          public vec4(vec4 v)
45          {
46              this.x = v.x;
47              this.y = v.y;
48              this.z = v.z;
49              this.w = v.w;
50          }
51
52          public vec4(vec3 xyz, float w)
53          {
54              this.x = xyz.x;
55              this.y = xyz.y;
56              this.z = xyz.z;
57              this.w = w;
58          }
59
60          public static vec4 operator +(vec4 lhs, vec4 rhs)
61          {
62              return new vec4(lhs.x + rhs.x, lhs.y + rhs.y, lhs.z + rhs.z, lhs.w + rh
s.w);
63          }
64
65          public static vec4 operator +(vec4 lhs, float rhs)
66          {
67              return new vec4(lhs.x + rhs, lhs.y + rhs, lhs.z + rhs, lhs.w + rhs);
68          }
69
70          public static vec4 operator -(vec4 lhs, float rhs)
71          {
72              return new vec4(lhs.x - rhs, lhs.y - rhs, lhs.z - rhs, lhs.w - rhs);
73          }
74
75          public static vec4 operator -(vec4 lhs, vec4 rhs)
76          {
77              return new vec4(lhs.x - rhs.x, lhs.y - rhs.y, lhs.z - rhs.z, lhs.w - rh
s.w);
```

评论排行榜


```
78          }
79
80          public static vec4 operator *(vec4 self, float s)
81          {
82              return new vec4(self.x * s, self.y * s, self.z * s, self.w * s);
83          }
84
85          public static vec4 operator *(float lhs, vec4 rhs)
86          {
87              return new vec4(rhs.x * lhs, rhs.y * lhs, rhs.z * lhs, rhs.w * lhs);
88          }
89
90          public static vec4 operator *(vec4 lhs, vec4 rhs)
91          {
92              return new vec4(rhs.x * lhs.x, rhs.y * lhs.y, rhs.z * lhs.z, rhs.w * lh
s.w);
93          }
94
95          public static vec4 operator /(vec4 lhs, float rhs)
96          {
97              return new vec4(lhs.x / rhs, lhs.y / rhs, lhs.z / rhs, lhs.w / rhs);
98          }
99
100         public float[] to_array()
101         {
102             return new[] { x, y, z, w };
103         }
104
105         /// <summary>
106         /// 归一化向量
107         /// </summary>
108         /// <param name="vector"></param>
109         /// <returns></returns>
110         public void Normalize()
111         {
112             var frt = (float)Math.Sqrt(this.x * this.x + this.y * this.y + this.z *
this.z);
113
114             this.x = x / frt;
115             this.y = y / frt;
116             this.z = z / frt;
117             this.w = w / frt;
118         }
119
120         public override string ToString()
121         {
122             return string.Format("{0:0.00},{1:0.00},{2:0.00},{3:0.00}", x, y, z,
w);
123         }
124     }
```

四维矩阵

然后，我们定义一个四维矩阵mat4。它用4个vec4表示，每个vec4代表一个列向量。（这是glm中的定义）

```
0
```

15. SharpGL(Opengl)入门之纹理星球(18)

```
/// <summary>
/// Represents a 4x4 matrix.
/// </summary>
public struct mat4
{
    public override string ToString()
    {
        if (cols == null)
        { return "<null>"; }
        var builder = new System.Text.StringBuilder();
        for (int i = 0; i < cols.Length; i++)
        {
            builder.Append(cols[i]);
            builder.Append(" + ");
        }
        return builder.ToString();
        //return base.ToString();
    }
    #region Construction

    /// <summary>
    /// Initializes a new instance of the <see cref="mat4"/> struct.
    /// This matrix is the identity matrix scaled by <paramref name="scale"/>.
    /// </summary>
    /// <param name="scale">The scale.</param>
    public mat4(float scale)
    {
        cols = new[]
        {
            new vec4(scale, 0.0f, 0.0f, 0.0f),
            new vec4(0.0f, scale, 0.0f, 0.0f),
            new vec4(0.0f, 0.0f, scale, 0.0f),
            new vec4(0.0f, 0.0f, 0.0f, scale),
        };
    }

    /// <summary>
    /// Initializes a new instance of the <see cref="mat4"/> struct.
    /// The matrix is initialised with the <paramref name="cols"/>.
    /// </summary>
    /// <param name="cols">The colums of the matrix.</param>
    public mat4(vec4[] cols)
    {
        this.cols = new[] { cols[0], cols[1], cols[2], cols[3] };
    }

    public mat4(vec4 a, vec4 b, vec4 c, vec4 d)
    {
        this.cols = new[]
        {
            a, b, c, d
        };
    }

    /// <summary>
    /// Creates an identity matrix.
    /// </summary>
    /// <returns>A new identity matrix.</returns>
    public static mat4 identity()
    {
```

```
61              return new mat4
62              {
63                  cols = new[]
64                  {
65                      new vec4(1,0,0,0),
66                      new vec4(0,1,0,0),
67                      new vec4(0,0,1,0),
68                      new vec4(0,0,0,1)
69                  }
70              };
71          }
72
73          #endregion
74
75          #region Index Access
76
77          /// <summary>
78          /// Gets or sets the <see cref="vec4"/> column at the specified index.
79          /// </summary>
80          /// <value>
81          /// The <see cref="vec4"/> column.
82          /// </value>
83          /// <param name="column">The column index.</param>
84          /// <returns>The column at index <paramref name="column"/>.</returns>
85          public vec4 this[int column]
86          {
87              get { return cols[column]; }
88              set { cols[column] = value; }
89          }
90
91          /// <summary>
92          /// Gets or sets the element at <paramref name="column"/> and <paramref nam
e="row"/>.
93          /// </summary>
94          /// <value>
95          /// The element at <paramref name="column"/> and <paramref name="row"/>.
96          /// </value>
97          /// <param name="column">The column index.</param>
98          /// <param name="row">The row index.</param>
99          /// <returns>
100         /// The element at <paramref name="column"/> and <paramref name="row"/>.
101         /// </returns>
102         public float this[int column, int row]
103         {
104             get { return cols[column][row]; }
105             set { cols[column][row] = value; }
106         }
107
108         #endregion
109
110         #region Conversion
111
112         /// <summary>
113         /// Returns the matrix as a flat array of elements, column major.
114         /// </summary>
115         /// <returns></returns>
116         public float[] to_array()
117         {
118             return cols.SelectMany(v => v.to_array()).ToArray();
119         }
```

```
120
121         /// <summary>
122         /// Returns the <see cref="mat3"/> portion of this matrix.
123         /// </summary>
124         /// <returns>The <see cref="mat3"/> portion of this matrix.</returns>
125         public mat3 to_mat3()
126         {
127             return new mat3(new[] {
128             new vec3(cols[0][0], cols[0][1], cols[0][2]),
129             new vec3(cols[1][0], cols[1][1], cols[1][2]),
130             new vec3(cols[2][0], cols[2][1], cols[2][2])});
131         }
132
133         #endregion
134
135         #region Multiplication
136
137         /// <summary>
138         /// Multiplies the <paramref name="lhs"/> matrix by the <paramref name="rh
s"/> vector.
139         /// </summary>
140         /// <param name="lhs">The LHS matrix.</param>
141         /// <param name="rhs">The RHS vector.</param>
142         /// <returns>The product of <paramref name="lhs"/> and <paramref name="rh
s"/>.</returns>
143         public static vec4 operator *(mat4 lhs, vec4 rhs)
144         {
145             return new vec4(
146                 lhs[0, 0] * rhs[0] + lhs[1, 0] * rhs[1] + lhs[2, 0] * rhs[2] + lhs
[3, 0] * rhs[3],
147                 lhs[0, 1] * rhs[0] + lhs[1, 1] * rhs[1] + lhs[2, 1] * rhs[2] + lhs
[3, 1] * rhs[3],
148                 lhs[0, 2] * rhs[0] + lhs[1, 2] * rhs[1] + lhs[2, 2] * rhs[2] + lhs
[3, 2] * rhs[3],
149                 lhs[0, 3] * rhs[0] + lhs[1, 3] * rhs[1] + lhs[2, 3] * rhs[2] + lhs
[3, 3] * rhs[3]
150             );
151         }
152
153         /// <summary>
154         /// Multiplies the <paramref name="lhs"/> matrix by the <paramref name="rh
s"/> matrix.
155         /// </summary>
156         /// <param name="lhs">The LHS matrix.</param>
157         /// <param name="rhs">The RHS matrix.</param>
158         /// <returns>The product of <paramref name="lhs"/> and <paramref name="rh
s"/>.</returns>
159         public static mat4 operator *(mat4 lhs, mat4 rhs)
160         {
161             mat4 result = new mat4(
162                 new vec4(
163                     lhs[0][0] * rhs[0][0] + lhs[1][0] * rhs[0][1] + lhs[2][0] * rhs
[0][2] + lhs[3][0] * rhs[0][3],
164                     lhs[0][1] * rhs[0][0] + lhs[1][1] * rhs[0][1] + lhs[2][1] * rhs
[0][2] + lhs[3][1] * rhs[0][3],
165                     lhs[0][2] * rhs[0][0] + lhs[1][2] * rhs[0][1] + lhs[2][2] * rhs
[0][2] + lhs[3][2] * rhs[0][3],
166                     lhs[0][3] * rhs[0][0] + lhs[1][3] * rhs[0][1] + lhs[2][3] * rhs
[0][2] + lhs[3][3] * rhs[0][3]
167                     ),
```

```
168                 new vec4(
169                     lhs[0][0] * rhs[1][0] + lhs[1][0] * rhs[1][1] + lhs[2][0] * rhs
[1][2] + lhs[3][0] * rhs[1][3],
170                     lhs[0][1] * rhs[1][0] + lhs[1][1] * rhs[1][1] + lhs[2][1] * rhs
[1][2] + lhs[3][1] * rhs[1][3],
171                     lhs[0][2] * rhs[1][0] + lhs[1][2] * rhs[1][1] + lhs[2][2] * rhs
[1][2] + lhs[3][2] * rhs[1][3],
172                     lhs[0][3] * rhs[1][0] + lhs[1][3] * rhs[1][1] + lhs[2][3] * rhs
[1][2] + lhs[3][3] * rhs[1][3]
173                     ),
174                 new vec4(
175                     lhs[0][0] * rhs[2][0] + lhs[1][0] * rhs[2][1] + lhs[2][0] * rhs
[2][2] + lhs[3][0] * rhs[2][3],
176                     lhs[0][1] * rhs[2][0] + lhs[1][1] * rhs[2][1] + lhs[2][1] * rhs
[2][2] + lhs[3][1] * rhs[2][3],
177                     lhs[0][2] * rhs[2][0] + lhs[1][2] * rhs[2][1] + lhs[2][2] * rhs
[2][2] + lhs[3][2] * rhs[2][3],
178                     lhs[0][3] * rhs[2][0] + lhs[1][3] * rhs[2][1] + lhs[2][3] * rhs
[2][2] + lhs[3][3] * rhs[2][3]
179                     ),
180                 new vec4(
181                     lhs[0][0] * rhs[3][0] + lhs[1][0] * rhs[3][1] + lhs[2][0] * rhs
[3][2] + lhs[3][0] * rhs[3][3],
182                     lhs[0][1] * rhs[3][0] + lhs[1][1] * rhs[3][1] + lhs[2][1] * rhs
[3][2] + lhs[3][1] * rhs[3][3],
183                     lhs[0][2] * rhs[3][0] + lhs[1][2] * rhs[3][1] + lhs[2][2] * rhs
[3][2] + lhs[3][2] * rhs[3][3],
184                     lhs[0][3] * rhs[3][0] + lhs[1][3] * rhs[3][1] + lhs[2][3] * rhs
[3][2] + lhs[3][3] * rhs[3][3]
185                     )
186                     );
187 
188             return result;
189         }
190 
191         public static mat4 operator *(mat4 lhs, float s)
192         {
193             return new mat4(new[]
194             {
195                 lhs[0]*s,
196                 lhs[1]*s,
197                 lhs[2]*s,
198                 lhs[3]*s
199             });
200         }
201 
202         #endregion
203 
204         /// <summary>
205         /// The columms of the matrix.
206         /// </summary>
207         private vec4[] cols;
208     }
```

回到顶部(go to top)

从ortho到矩阵

根据传入的参数可以获得一个代表平行投影的矩阵。

```
        /// <summary>
        /// Creates a matrix for an orthographic parallel viewing volume.
        /// </summary>
        /// <param name="left">The left.</param>
        /// <param name="right">The right.</param>
        /// <param name="bottom">The bottom.</param>
        /// <param name="top">The top.</param>
        /// <param name="zNear">The z near.</param>
        /// <param name="zFar">The z far.</param>
        /// <returns></returns>
        public static mat4 ortho(float left, float right, float bottom, float top, f
loat zNear, float zFar)
        {
            var result = mat4.identity();
            result[0, 0] = (2f) / (right - left);
            result[1, 1] = (2f) / (top - bottom);
            result[2, 2] = -(2f) / (zFar - zNear);
            result[3, 0] = -(right + left) / (right - left);
            result[3, 1] = -(top + bottom) / (top - bottom);
            result[3, 2] = -(zFar + zNear) / (zFar - zNear);
            return result;
        }
```

从矩阵到ortho

反过来，当我们手上有一个矩阵时，我们可以分析出这个矩阵是由ortho用怎样的参数计算得到的。（当然，并非所有矩阵都能用ortho计算出来）

```
        /// <summary>
        /// 如果此矩阵是glm.ortho()的结果，那么返回glm.ortho()的各个参数值。
        /// </summary>
        /// <param name="matrix"></param>
        /// <param name="left"></param>
        /// <param name="right"></param>
        /// <param name="bottom"></param>
        /// <param name="top"></param>
        /// <param name="zNear"></param>
        /// <param name="zFar"></param>
        /// <returns></returns>
        public static bool TryParse(this mat4 matrix,
            out float left, out float right, out float bottom, out float top, out fl
oat zNear, out float zFar)
        {
            {
                float negHalfLeftRight = matrix[3, 0] / matrix[0, 0];
                float halfRightMinusLeft = 1.0f / matrix[0][0];
                left = -(halfRightMinusLeft + negHalfLeftRight);
                right = halfRightMinusLeft - negHalfLeftRight;
            }

            {
                float negHalfBottomTop = matrix[3, 1] / matrix[1, 1];
                float halfTopMinusBottom = 1.0f / matrix[1, 1];
```

```
                bottom = -(halfTopMinusBottom + negHalfBottomTop);
                top = halfTopMinusBottom - negHalfBottomTop;
            }

            {
                float halfNearFar = matrix[3, 2] / matrix[2, 2];
                float negHalfFarMinusNear = 1.0f / matrix[2, 2];
                zNear = negHalfFarMinusNear + halfNearFar;
                zFar = halfNearFar - negHalfFarMinusNear;
            }

            if (matrix[0, 0] == 0.0f || matrix[1, 1] == 0.0f || matrix[2, 2] == 0.0
f)
            {
                return false;
            }

            if (matrix[1, 0] != 0.0f || matrix[2, 0] != 0.0f
                || matrix[0, 1] != 0.0f || matrix[2, 1] != 0.0f
                || matrix[0, 2] != 0.0f || matrix[1, 2] != 0.0f
                || matrix[0, 3] != 0.0f || matrix[1, 3] != 0.0f || matrix[2, 3] !=
0.0f)
            {
                return false;
            }

            if (matrix[3, 3] != 1.0f)
            {
                return false;
            }

            return true;
        }
```

回到顶部(go to top)

矩阵与perpspective的转换

从perspective到矩阵

根据传入的参数可以获得一个代表透视投影的矩阵。

```
        /// <summary>
        /// Creates a perspective transformation matrix.
        /// </summary>
        /// <param name="fovy">The field of view angle, in radians.</param>
        /// <param name="aspect">The aspect ratio.</param>
        /// <param name="zNear">The near depth clipping plane.</param>
        /// <param name="zFar">The far depth clipping plane.</param>
        /// <returns>A <see cref="mat4"/> that contains the projection matrix for th
e perspective transformation.</returns>
        public static mat4 perspective(float fovy, float aspect, float zNear, float
zFar)
        {
            var result = mat4.identity();
            float tangent = (float)Math.Tan(fovy / 2.0f);
            float height = zNear * tangent;
```

O

```
            float width = height * aspect;
            float l = -width, r = width, b = -height, t = height, n = zNear, f = zFa
r;
            result[0, 0] = 2.0f * n / (r - l);// = 2.0f * zNear / (2.0f * zNear * ta
ngent * aspect)
            result[1, 1] = 2.0f * n / (t - b);// = 2.0f * zNear / (2.0f * zNear * ta
ngent)
            //result[2, 0] = (r + l) / (r - l);// = 0.0f
            //result[2, 1] = (t + b) / (t - b);// = 0.0f
            result[2, 2] = -(f + n) / (f - n);
            result[2, 3] = -1.0f;
            result[3, 2] = -(2.0f * f * n) / (f - n);
            result[3, 3] = 0.0f;

            return result;
        }
```

从矩阵到perspective

反过来，当我们手上有一个矩阵时，我们可以分析出这个矩阵是由perpspective用怎样的参数计算得到的。（当然，并非所有矩阵都能用perpspective计算出来）

```
        /// <summary>
        /// 如果此矩阵是glm.perspective()的结果，那么返回glm.perspective()的各个参数值。
        /// </summary>
        /// <param name="matrix"></param>
        /// <param name="fovy"></param>
        /// <param name="aspectRatio"></param>
        /// <param name="zNear"></param>
        /// <param name="zFar"></param>
        /// <returns></returns>
        public static bool TryParse(this mat4 matrix,
            out float fovy, out float aspectRatio, out float zNear, out float zFar)
        {
            float tanHalfFovy = 1.0f / matrix[1, 1];
            fovy = 2 * (float)(Math.Atan(tanHalfFovy));
            if (fovy < 0) { fovy = -fovy; }
            //aspectRatio = 1.0f / matrix[0, 0] / tanHalfFovy;
            aspectRatio = matrix[1, 1] / matrix[0, 0];
            if (matrix[2, 2] == 1.0f)
            {
                zFar = 0.0f;
                zNear = 0.0f;
            }
            else if (matrix[2, 2] == -1.0f)
            {
                zNear = 0.0f;
                zFar = float.PositiveInfinity;
            }
            else
            {
                zNear = matrix[3, 2] / (matrix[2, 2] - 1);
                zFar = matrix[3, 2] / (matrix[2, 2] + 1);
            }

            if (matrix[0, 0] == 0.0f || matrix[1, 1] == 0.0f || matrix[2, 2] == 0.0
f)
```

O

```
35              {
36                  return false;
37              }
38
39              if (matrix[1, 0] != 0.0f || matrix[3, 0] != 0.0f
40                  || matrix[0, 1] != 0.0f || matrix[3, 1] != 0.0f
41                  || matrix[0, 2] != 0.0f || matrix[1, 2] != 0.0f
42                  || matrix[0, 3] != 0.0f || matrix[1, 3] != 0.0f || matrix[3, 3] !=
0.0f)
43              {
44                  return false;
45              }
46
47              if (matrix[2, 3] != -1.0f)
48              {
49                  return false;
50              }
51
52              return true;
53          }
```

回到顶部(go to top)

总结

本篇就写这些，今后再写一些相关的内容。

如果您愿意花几块钱请我喝杯茶的话，可以用手机扫描下方的二维码，通过微信捐赠。我会努力写出更好的文章。
（微信捐赠不显示捐赠者的个人信息，如需要，请注明您的联系方式（微信留言只显示10个汉字））
Thank you for your kindly donation!

微信捐赠二维码:
Donate by microMsg:

标签: opengl

BIT祝威
关注 - 12
粉丝 - 765
推荐博客
+加关注

« 上一篇: C#+OpenGL+FreeType显示3D文字(3) - 用PointSprite绘制文字
» 下一篇: CSharpGL(0)一个易学易用的C#版OpenGL

posted @ 2015-08-27 00:38 BIT祝威 阅读(2754) 评论(0) 编辑 收藏

Large Visitor Globe