

天下事有难易乎

有

博客园

首页

新随笔

联系

管理

Only Title | Show Abstract 随笔 - 226 文章 - 0 评论 - 1366



靡不有初，鲜克有终



CSharpGL @ GitHub



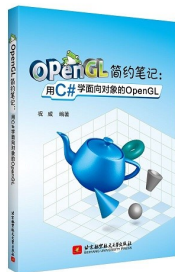
我的博客汇总



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)

- 定义矩阵
 - 四维向量
 - 四维矩阵
- 矩阵与ortho的转换
 - 从ortho到矩阵
 - 从矩阵到ortho
- 矩阵与perspective的转换
 - 从perspective到矩阵
 - 从矩阵到perspective
- 总结

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。

```
1  /// <summary>
2  /// Represents a four dimensional vector.
3  /// </summary>
4  public struct vec4
5  {
6      public float x;
7      public float y;
8      public float z;
9      public float w;
10
11     public float this[int index]
12     {
13         get
14         {
15             if (index == 0) return x;
16             else if (index == 1) return y;
17             else if (index == 2) return z;
18             else if (index == 3) return w;
19             else throw new Exception("Out of range.");
20         }
21     }
22 }
```

2. Re:CSHarpGL(28)得到高精度可定制字形贴图的极简方法

@ 夜、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 + rhs.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 - rhs.w);
78 }
```

10. [Unity3D入门]分享一个自制的入门级游戏项目"坦克狙击手"(10236)

11. SharpGL(OpenGL)入门之纹理星球(10212)

12. 网段，子网掩码，网络标识，IP划分(9332)

13. 【OpenGL(SharpGL)】支持任意相机可平移缩放的轨迹球实现(8685)

14. 一个编译器的实现0(8656)

15. 《30天自制操作系统》笔记(12)——多任务入门(8099)

评论排行榜

1. C#自定义控件：WinForm将其它应用程序窗口嵌入自己内部(195)

2. 继电器是如何成为CPU的 (1) (169)

3. [Unity3D入门]入门级游戏项目"坦克狙击手"更新(104)

4. [Unity3D入门]分享一个自制的入门级游戏项目"坦克狙击手"(103)

5. CSharpGL(0)一个易学易用的C#版OpenGL(49)

6. [我给Unity官方视频教程做中文字幕]beginner Graphics – Lessons系列之摄像机介绍Cameras(39)

7. [我给Unity官方视频教程做中文字幕]beginner Graphics – Lessons系列之网格渲染器和过滤器Mesh renderers and filters(33)

8. 继电器是如何成为CPU的 (2) (33)

9. 《30天自制操作系统》笔记(02)——导入C语言(29)

10. 《30天自制操作系统》笔记(01)——hello bitzhuwei's OS!(27)

11. [我给Unity官方视频教程做中文字幕]beginner Graphics – Lessons系列之材质了解Materials(24)

12. 用C表达面向对象语言的机制——C#版(24)

13. [我给Unity官方视频教程做中文字幕]beginner Graphics – Lessons系列之纹理Textures(23)

14. [我给Unity官方视频教程做中文字幕]beginner Graphics – Lessons系列之灯光介绍Lights(18)

```
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 * lhs.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

```
1  /// <summary>
2  /// Represents a 4x4 matrix.
3  /// </summary>
4  public struct mat4
5  {
6      public override string ToString()
7      {
8          if (cols == null)
9          { return "<null>"; }
10         var builder = new System.Text.StringBuilder();
11         for (int i = 0; i < cols.Length; i++)
12         {
13             builder.Append(cols[i]);
14             builder.Append(" + ");
15         }
16         return builder.ToString();
17         //return base.ToString();
18     }
19     #region Construction
20
21     /// <summary>
22     /// Initializes a new instance of the <see cref="mat4"/> struct.
23     /// This matrix is the identity matrix scaled by <paramref name="scale"/>.
24     /// </summary>
25     /// <param name="scale">The scale.</param>
26     public mat4(float scale)
27     {
28         cols = new[]
29         {
30             new vec4(scale, 0.0f, 0.0f, 0.0f),
31             new vec4(0.0f, scale, 0.0f, 0.0f),
32             new vec4(0.0f, 0.0f, scale, 0.0f),
33             new vec4(0.0f, 0.0f, 0.0f, scale),
34         };
35     }
36
37     /// <summary>
38     /// Initializes a new instance of the <see cref="mat4"/> struct.
39     /// The matrix is initialised with the <paramref name="cols"/>.
40     /// </summary>
41     /// <param name="cols">The columns of the matrix.</param>
42     public mat4(vec4[] cols)
43     {
44         this.cols = new[] { cols[0], cols[1], cols[2], cols[3] };
45     }
46
47     public mat4(vec4 a, vec4 b, vec4 c, vec4 d)
48     {
49         this.cols = new[]
50         {
51             a, b, c, d
52         };
53     }
54
55     /// <summary>
56     /// Creates an identity matrix.
57     /// </summary>
58     /// <returns>A new identity matrix.</returns>
59     public static mat4 identity()
60     {
```

```

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
139 s"/> vector.
140     /// </summary>
141     /// <param name="lhs">The LHS matrix.</param>
142     /// <param name="rhs">The RHS vector.</param>
143     /// <returns>The product of <paramref name="lhs"/> and <paramref name="rh
144 s"/>.</returns>
145     public static vec4 operator *(mat4 lhs, vec4 rhs)
146     {
147         return new vec4(
148             lhs[0, 0] * rhs[0] + lhs[1, 0] * rhs[1] + lhs[2, 0] * rhs[2] + lhs
149 [3, 0] * rhs[3],
150             lhs[0, 1] * rhs[0] + lhs[1, 1] * rhs[1] + lhs[2, 1] * rhs[2] + lhs
151 [3, 1] * rhs[3],
152             lhs[0, 2] * rhs[0] + lhs[1, 2] * rhs[1] + lhs[2, 2] * rhs[2] + lhs
153 [3, 2] * rhs[3],
154             lhs[0, 3] * rhs[0] + lhs[1, 3] * rhs[1] + lhs[2, 3] * rhs[2] + lhs
155 [3, 3] * rhs[3]
156         );
157     }
158
159     /// <summary>
160     /// Multiplies the <paramref name="lhs"/> matrix by the <paramref name="rh
161 s"/> matrix.
162     /// </summary>
163     /// <param name="lhs">The LHS matrix.</param>
164     /// <param name="rhs">The RHS matrix.</param>
165     /// <returns>The product of <paramref name="lhs"/> and <paramref name="rh
166 s"/>.</returns>
167     public static mat4 operator *(mat4 lhs, mat4 rhs)
168     {
169         mat4 result = new mat4(
170             new vec4(
171                 lhs[0][0] * rhs[0][0] + lhs[1][0] * rhs[0][1] + lhs[2][0] * rhs
172 [0][2] + lhs[3][0] * rhs[0][3],
173                 lhs[0][1] * rhs[0][0] + lhs[1][1] * rhs[0][1] + lhs[2][1] * rhs
174 [0][2] + lhs[3][1] * rhs[0][3],
175                 lhs[0][2] * rhs[0][0] + lhs[1][2] * rhs[0][1] + lhs[2][2] * rhs
176 [0][2] + lhs[3][2] * rhs[0][3],
177                 lhs[0][3] * rhs[0][0] + lhs[1][3] * rhs[0][1] + lhs[2][3] * rhs
178 [0][2] + lhs[3][3] * rhs[0][3]
179             ),
180             new vec4(
181                 lhs[0][0] * rhs[1][0] + lhs[1][0] * rhs[1][1] + lhs[2][0] * rhs
182 [1][2] + lhs[3][0] * rhs[1][3],
183                 lhs[0][1] * rhs[1][0] + lhs[1][1] * rhs[1][1] + lhs[2][1] * rhs
184 [1][2] + lhs[3][1] * rhs[1][3],
185                 lhs[0][2] * rhs[1][0] + lhs[1][2] * rhs[1][1] + lhs[2][2] * rhs
186 [1][2] + lhs[3][2] * rhs[1][3],
187                 lhs[0][3] * rhs[1][0] + lhs[1][3] * rhs[1][1] + lhs[2][3] * rhs
188 [1][2] + lhs[3][3] * rhs[1][3]
189             ),
190             new vec4(
191                 lhs[0][0] * rhs[2][0] + lhs[1][0] * rhs[2][1] + lhs[2][0] * rhs
192 [2][2] + lhs[3][0] * rhs[2][3],
193                 lhs[0][1] * rhs[2][0] + lhs[1][1] * rhs[2][1] + lhs[2][1] * rhs
194 [2][2] + lhs[3][1] * rhs[2][3],
195                 lhs[0][2] * rhs[2][0] + lhs[1][2] * rhs[2][1] + lhs[2][2] * rhs
196 [2][2] + lhs[3][2] * rhs[2][3],
197                 lhs[0][3] * rhs[2][0] + lhs[1][3] * rhs[2][1] + lhs[2][3] * rhs
198 [2][2] + lhs[3][3] * rhs[2][3]
199             ),
200             new vec4(
201                 lhs[0][0] * rhs[3][0] + lhs[1][0] * rhs[3][1] + lhs[2][0] * rhs
202 [3][2] + lhs[3][0] * rhs[3][3],
203                 lhs[0][1] * rhs[3][0] + lhs[1][1] * rhs[3][1] + lhs[2][1] * rhs
204 [3][2] + lhs[3][1] * rhs[3][3],
205                 lhs[0][2] * rhs[3][0] + lhs[1][2] * rhs[3][1] + lhs[2][2] * rhs
206 [3][2] + lhs[3][2] * rhs[3][3],
207                 lhs[0][3] * rhs[3][0] + lhs[1][3] * rhs[3][1] + lhs[2][3] * rhs
208 [3][2] + lhs[3][3] * rhs[3][3]
209             )
210         );
211     }
212
213     #endregion
214
215     #region Vector operations
216
217     /// <summary>
218     /// Adds the <paramref name="lhs"/> vector to the <paramref name="rhs"/> vector.
219     /// </summary>
220     /// <param name="lhs">The LHS vector.</param>
221     /// <param name="rhs">The RHS vector.</param>
222     /// <returns>The sum of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
223     public static vec4 operator +(vec4 lhs, vec4 rhs)
224     {
225         return new vec4(
226             lhs[0] + rhs[0],
227             lhs[1] + rhs[1],
228             lhs[2] + rhs[2],
229             lhs[3] + rhs[3]
230         );
231     }
232
233     /// <summary>
234     /// Subtracts the <paramref name="lhs"/> vector from the <paramref name="rhs"/> vector.
235     /// </summary>
236     /// <param name="lhs">The LHS vector.</param>
237     /// <param name="rhs">The RHS vector.</param>
238     /// <returns>The difference of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
239     public static vec4 operator -(vec4 lhs, vec4 rhs)
240     {
241         return new vec4(
242             lhs[0] - rhs[0],
243             lhs[1] - rhs[1],
244             lhs[2] - rhs[2],
245             lhs[3] - rhs[3]
246         );
247     }
248
249     /// <summary>
250     /// Multiplies the <paramref name="lhs"/> vector by the <paramref name="rhs"/> scalar.
251     /// </summary>
252     /// <param name="lhs">The LHS vector.</param>
253     /// <param name="rhs">The RHS scalar.</param>
254     /// <returns>The product of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
255     public static vec4 operator *(vec4 lhs, double rhs)
256     {
257         return new vec4(
258             lhs[0] * rhs,
259             lhs[1] * rhs,
260             lhs[2] * rhs,
261             lhs[3] * rhs
262         );
263     }
264
265     /// <summary>
266     /// Divides the <paramref name="lhs"/> vector by the <paramref name="rhs"/> scalar.
267     /// </summary>
268     /// <param name="lhs">The LHS vector.</param>
269     /// <param name="rhs">The RHS scalar.</param>
270     /// <returns>The quotient of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
271     public static vec4 operator /(vec4 lhs, double rhs)
272     {
273         return new vec4(
274             lhs[0] / rhs,
275             lhs[1] / rhs,
276             lhs[2] / rhs,
277             lhs[3] / rhs
278         );
279     }
280
281     #endregion
282
283     #region Matrix operations
284
285     /// <summary>
286     /// Transposes the <paramref name="matrix"/> matrix.
287     /// </summary>
288     /// <param name="matrix">The matrix to be transposed.</param>
289     /// <returns>The transposed matrix.</returns>
290     public static mat4 operator T(mat4 matrix)
291     {
292         return new mat4(
293             new vec4(
294                 matrix[0][0], matrix[1][0], matrix[2][0], matrix[3][0],
295                 matrix[0][1], matrix[1][1], matrix[2][1], matrix[3][1],
296                 matrix[0][2], matrix[1][2], matrix[2][2], matrix[3][2],
297                 matrix[0][3], matrix[1][3], matrix[2][3], matrix[3][3]
298             )
299         );
300     }
301
302     /// <summary>
303     /// Inverts the <paramref name="matrix"/> matrix.
304     /// </summary>
305     /// <param name="matrix">The matrix to be inverted.</param>
306     /// <returns>The inverted matrix.</returns>
307     public static mat4 operator I(mat4 matrix)
308     {
309         return new mat4(
310             new vec4(
311                 matrix[0][0], matrix[1][0], matrix[2][0], matrix[3][0],
312                 matrix[0][1], matrix[1][1], matrix[2][1], matrix[3][1],
313                 matrix[0][2], matrix[1][2], matrix[2][2], matrix[3][2],
314                 matrix[0][3], matrix[1][3], matrix[2][3], matrix[3][3]
315             )
316         );
317     }
318
319     #endregion
320
321     #region Utility methods
322
323     /// <summary>
324     /// Returns the determinant of the <paramref name="matrix"/> matrix.
325     /// </summary>
326     /// <param name="matrix">The matrix to be evaluated.</param>
327     /// <returns>The determinant of the matrix.</returns>
328     public static double Determinant(mat4 matrix)
329     {
330         return 0;
331     }
332
333     #endregion
334
335     #region Constants
336
337     /// <summary>
338     /// The identity matrix.
339     /// </summary>
340     public static mat4 Identity
341     {
342         get { return new mat4(
343             new vec4(
344                 1, 0, 0, 0,
345                 0, 1, 0, 0,
346                 0, 0, 1, 0,
347                 0, 0, 0, 1
348             )
349         ); }
350     }
351
352     #endregion
353
354     #region Constructors
355
356     /// <summary>
357     /// Constructs a new matrix from the given columns.
358     /// </summary>
359     /// <param name="cols">The columns of the matrix.</param>
360     public mat3(vec3[] cols)
361     {
362         this.cols = cols;
363     }
364
365     /// <summary>
366     /// Constructs a new matrix from the given rows.
367     /// </summary>
368     /// <param name="rows">The rows of the matrix.</param>
369     public mat3(vec3[] rows)
370     {
371         this.rows = rows;
372     }
373
374     #endregion
375
376     #region Properties
377
378     /// <summary>
379     /// The columns of the matrix.
380     /// </summary>
381     public vec3[] cols
382     {
383         get;
384     }
385
386     /// <summary>
387     /// The rows of the matrix.
388     /// </summary>
389     public vec3[] rows
390     {
391         get;
392     }
393
394     #endregion
395
396     #region Operators
397
398     /// <summary>
399     /// Adds the <paramref name="lhs"/> matrix to the <paramref name="rhs"/> matrix.
400     /// </summary>
401     /// <param name="lhs">The LHS matrix.</param>
402     /// <param name="rhs">The RHS matrix.</param>
403     /// <returns>The sum of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
404     public static mat3 operator +(mat3 lhs, mat3 rhs)
405     {
406         return new mat3(new vec3[] {
407             new vec3(lhs.cols[0][0] + rhs.cols[0][0], lhs.cols[0][1] + rhs.cols[0][1], lhs.cols[0][2] + rhs.cols[0][2]),
408             new vec3(lhs.cols[1][0] + rhs.cols[1][0], lhs.cols[1][1] + rhs.cols[1][1], lhs.cols[1][2] + rhs.cols[1][2]),
409             new vec3(lhs.cols[2][0] + rhs.cols[2][0], lhs.cols[2][1] + rhs.cols[2][1], lhs.cols[2][2] + rhs.cols[2][2])
410         });
411     }
412
413     /// <summary>
414     /// Subtracts the <paramref name="lhs"/> matrix from the <paramref name="rhs"/> matrix.
415     /// </summary>
416     /// <param name="lhs">The LHS matrix.</param>
417     /// <param name="rhs">The RHS matrix.</param>
418     /// <returns>The difference of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
419     public static mat3 operator -(mat3 lhs, mat3 rhs)
420     {
421         return new mat3(new vec3[] {
422             new vec3(lhs.cols[0][0] - rhs.cols[0][0], lhs.cols[0][1] - rhs.cols[0][1], lhs.cols[0][2] - rhs.cols[0][2]),
423             new vec3(lhs.cols[1][0] - rhs.cols[1][0], lhs.cols[1][1] - rhs.cols[1][1], lhs.cols[1][2] - rhs.cols[1][2]),
424             new vec3(lhs.cols[2][0] - rhs.cols[2][0], lhs.cols[2][1] - rhs.cols[2][1], lhs.cols[2][2] - rhs.cols[2][2])
425         });
426     }
427
428     /// <summary>
429     /// Multiplies the <paramref name="lhs"/> matrix by the <paramref name="rhs"/> scalar.
430     /// </summary>
431     /// <param name="lhs">The LHS matrix.</param>
432     /// <param name="rhs">The RHS scalar.</param>
433     /// <returns>The product of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
434     public static mat3 operator *(mat3 lhs, double rhs)
435     {
436         return new mat3(new vec3[] {
437             new vec3(lhs.cols[0][0] * rhs, lhs.cols[0][1] * rhs, lhs.cols[0][2] * rhs),
438             new vec3(lhs.cols[1][0] * rhs, lhs.cols[1][1] * rhs, lhs.cols[1][2] * rhs),
439             new vec3(lhs.cols[2][0] * rhs, lhs.cols[2][1] * rhs, lhs.cols[2][2] * rhs)
440         });
441     }
442
443     /// <summary>
444     /// Divides the <paramref name="lhs"/> matrix by the <paramref name="rhs"/> scalar.
445     /// </summary>
446     /// <param name="lhs">The LHS matrix.</param>
447     /// <param name="rhs">The RHS scalar.</param>
448     /// <returns>The quotient of <paramref name="lhs"/> and <paramref name="rhs"/>.</returns>
449     public static mat3 operator /(mat3 lhs, double rhs)
450     {
451         return new mat3(new vec3[] {
452             new vec3(lhs.cols[0][0] / rhs, lhs.cols[0][1] / rhs, lhs.cols[0][2] / rhs),
453             new vec3(lhs.cols[1][0] / rhs, lhs.cols[1][1] / rhs, lhs.cols[1][2] / rhs),
454             new vec3(lhs.cols[2][0] / rhs, lhs.cols[2][1] / rhs, lhs.cols[2][2] / rhs)
455         });
456     }
457
458     #endregion
459
460     #region Methods
461
462     /// <summary>
463     /// Returns the trace of the matrix.
464     /// </summary>
465     public double Trace()
466     {
467         return cols[0][0] + cols[1][1] + cols[2][2];
468     }
469
470     #endregion
471
472     #region Static Methods
473
474     /// <summary>
475     /// Returns the determinant of the matrix.
476     /// </summary>
477     public static double Determinant(mat3 matrix)
478     {
479         return 0;
480     }
481
482     #endregion
483
484     #region Private Methods
485
486     #endregion
487
488     #region Private Fields
489
490     #endregion
491
492     #region Private Constants
493
494     #endregion
495
496     #region Private Constructors
497
498     #endregion
499
500     #region Private Operators
501
502     #endregion
503
504     #region Private Properties
505
506     #endregion
507
508     #region Private Static Methods
509
510     #endregion
511
512     #region Private Static Properties
513
514     #endregion
515
516     #region Private Static Constants
517
518     #endregion
519
520     #region Private Static Constructors
521
522     #endregion
523
524     #region Private Static Operators
525
526     #endregion
527
528     #region Private Static Properties
529
530     #endregion
531
532     #region Private Static Constants
533
534     #endregion
535
536     #region Private Static Constructors
537
538     #endregion
539
540     #region Private Static Operators
541
542     #endregion
543
544     #region Private Static Properties
545
546     #endregion
547
548     #region Private Static Constants
549
550     #endregion
551
552     #region Private Static Constructors
553
554     #endregion
555
556     #region Private Static Operators
557
558     #endregion
559
560     #region Private Static Properties
561
562     #endregion
563
564     #region Private Static Constants
565
566     #endregion
567
568     #region Private Static Constructors
569
570     #endregion
571
572     #region Private Static Operators
573
574     #endregion
575
576     #region Private Static Properties
577
578     #endregion
579
580     #region Private Static Constants
581
582     #endregion
583
584     #region Private Static Constructors
585
586     #endregion
587
588     #region Private Static Operators
589
590     #endregion
591
592     #region Private Static Properties
593
594     #endregion
595
596     #region Private Static Constants
597
598     #endregion
599
600     #region Private Static Constructors
601
602     #endregion
603
604     #region Private Static Operators
605
606     #endregion
607
608     #region Private Static Properties
609
610     #endregion
611
612     #region Private Static Constants
613
614     #endregion
615
616     #region Private Static Constructors
617
618     #endregion
619
620     #region Private Static Operators
621
622     #endregion
623
624     #region Private Static Properties
625
626     #endregion
627
628     #region Private Static Constants
629
630     #endregion
631
632     #region Private Static Constructors
633
634     #endregion
635
636     #region Private Static Operators
637
638     #endregion
639
640     #region Private Static Properties
641
642     #endregion
643
644     #region Private Static Constants
645
646     #endregion
647
648     #region Private Static Constructors
649
650     #endregion
651
652     #region Private Static Operators
653
654     #endregion
655
656     #region Private Static Properties
657
658     #endregion
659
660     #region Private Static Constants
661
662     #endregion
663
664     #region Private Static Constructors
665
666     #endregion
667
668     #region Private Static Operators
669
670     #endregion
671
672     #region Private Static Properties
673
674     #endregion
675
676     #region Private Static Constants
677
678     #endregion
679
680     #region Private Static Constructors
681
682     #endregion
683
684     #region Private Static Operators
685
686     #endregion
687
688     #region Private Static Properties
689
690     #endregion
691
692     #region Private Static Constants
693
694     #endregion
695
696     #region Private Static Constructors
697
698     #endregion
699
700     #region Private Static Operators
701
702     #endregion
703
704     #region Private Static Properties
705
706     #endregion
707
708     #region Private Static Constants
709
710     #endregion
711
712     #region Private Static Constructors
713
714     #endregion
715
716     #region Private Static Operators
717
718     #endregion
719
720     #region Private Static Properties
721
722     #endregion
723
724     #region Private Static Constants
725
726     #endregion
727
728     #region Private Static Constructors
729
730     #endregion
731
732     #region Private Static Operators
733
734     #endregion
735
736     #region Private Static Properties
737
738     #endregion
739
740     #region Private Static Constants
741
742     #endregion
743
744     #region Private Static Constructors
745
746     #endregion
747
748     #region Private Static Operators
749
750     #endregion
751
752     #region Private Static Properties
753
754     #endregion
755
756     #region Private Static Constants
757
758     #endregion
759
760     #region Private Static Constructors
761
762     #endregion
763
764     #region Private Static Operators
765
766     #endregion
767
768     #region Private Static Properties
769
770     #endregion
771
772     #region Private Static Constants
773
774     #endregion
775
776     #region Private Static Constructors
777
778     #endregion
779
780     #region Private Static Operators
781
782     #endregion
783
784     #region Private Static Properties
785
786     #endregion
787
788     #region Private Static Constants
789
790     #endregion
791
792     #region Private Static Constructors
793
794     #endregion
795
796     #region Private Static Operators
797
798     #endregion
799
800     #region Private Static Properties
801
802     #endregion
803
804     #region Private Static Constants
805
806     #endregion
807
808     #region Private Static Constructors
809
810     #endregion
811
812     #region Private Static Operators
813
814     #endregion
815
816     #region Private Static Properties
817
818     #endregion
819
820     #region Private Static Constants
821
822     #endregion
823
824     #region Private Static Constructors
825
826     #endregion
827
828     #region Private Static Operators
829
830     #endregion
831
832     #region Private Static Properties
833
834     #endregion
835
836     #region Private Static Constants
837
838     #endregion
839
840     #region Private Static Constructors
841
842     #endregion
843
844     #region Private Static Operators
845
846     #endregion
847
848     #region Private Static Properties
849
850     #endregion
851
852     #region Private Static Constants
853
854     #endregion
855
856     #region Private Static Constructors
857
858     #endregion
859
860     #region Private Static Operators
861
862     #endregion
863
864     #region Private Static Properties
865
866     #endregion
867
868     #region Private Static Constants
869
870     #endregion
871
872     #region Private Static Constructors
873
874     #endregion
875
876     #region Private Static Operators
877
878     #endregion
879
880     #region Private Static Properties
881
882     #endregion
883
884     #region Private Static Constants
885
886     #endregion
887
888     #region Private Static Constructors
889
890     #endregion
891
892     #region Private Static Operators
893
894     #endregion
895
896     #region Private Static Properties
897
898     #endregion
899
900     #region Private Static Constants
901
902     #endregion
903
904     #region Private Static Constructors
905
906     #endregion
907
908     #region Private Static Operators
909
909 
```

```

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 columns of the matrix.
206 /// </summary>
207 private vec4[] cols;
208 }

```



[回到顶部\(go to top\)](#)

0

矩阵与ortho的转换

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。

```
1      /// <summary>
2      /// Creates a matrix for an orthographic parallel viewing volume.
3      /// </summary>
4      /// <param name="left">The left.</param>
5      /// <param name="right">The right.</param>
6      /// <param name="bottom">The bottom.</param>
7      /// <param name="top">The top.</param>
8      /// <param name="zNear">The z near.</param>
9      /// <param name="zFar">The z far.</param>
10     /// <returns></returns>
11     public static mat4 ortho(float left, float right, float bottom, float top, float zNear, float zFar)
12     {
13         var result = mat4.identity();
14         result[0, 0] = (2f) / (right - left);
15         result[1, 1] = (2f) / (top - bottom);
16         result[2, 2] = -(2f) / (zFar - zNear);
17         result[3, 0] = -(right + left) / (right - left);
18         result[3, 1] = -(top + bottom) / (top - bottom);
19         result[3, 2] = -(zFar + zNear) / (zFar - zNear);
20         return result;
21     }
```

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）

```
1      /// <summary>
2      /// 如果此矩阵是glm.ortho() 的结果，那么返回glm.ortho() 的各个参数值。
3      /// </summary>
4      /// <param name="matrix"></param>
5      /// <param name="left"></param>
6      /// <param name="right"></param>
7      /// <param name="bottom"></param>
8      /// <param name="top"></param>
9      /// <param name="zNear"></param>
10     /// <param name="zFar"></param>
11     /// <returns></returns>
12     public static bool TryParse(this mat4 matrix,
13         out float left, out float right, out float bottom, out float top, out float zNear, out float zFar)
14     {
15         {
16             float negHalfLeftRight = matrix[3, 0] / matrix[0, 0];
17             float halfRightMinusLeft = 1.0f / matrix[0][0];
18             left = -(halfRightMinusLeft + negHalfLeftRight);
19             right = halfRightMinusLeft - negHalfLeftRight;
20         }
21
22         {
23             float negHalfBottomTop = matrix[3, 1] / matrix[1, 1];
24             float halfTopMinusBottom = 1.0f / matrix[1, 1];
```



```

25         bottom = -(halfTopMinusBottom + negHalfBottomTop);
26         top = halfTopMinusBottom - negHalfBottomTop;
27     }
28
29     {
30         float halfNearFar = matrix[3, 2] / matrix[2, 2];
31         float negHalfFarMinusNear = 1.0f / matrix[2, 2];
32         zNear = negHalfFarMinusNear + halfNearFar;
33         zFar = halfNearFar - negHalfFarMinusNear;
34     }
35
36     if (matrix[0, 0] == 0.0f || matrix[1, 1] == 0.0f || matrix[2, 2] == 0.0
f)
37     {
38         return false;
39     }
40
41     if (matrix[1, 0] != 0.0f || matrix[2, 0] != 0.0f
42         || matrix[0, 1] != 0.0f || matrix[2, 1] != 0.0f
43         || matrix[0, 2] != 0.0f || matrix[1, 2] != 0.0f
44         || matrix[0, 3] != 0.0f || matrix[1, 3] != 0.0f || matrix[2, 3] !=
0.0f)
45     {
46         return false;
47     }
48
49     if (matrix[3, 3] != 1.0f)
50     {
51         return false;
52     }
53
54     return true;
55 }

```

[回到顶部\(go to top\)](#)

矩阵与perspective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。

```

1      /// <summary>
2      /// Creates a perspective transformation matrix.
3      /// </summary>
4      /// <param name="fovy">The field of view angle, in radians.</param>
5      /// <param name="aspect">The aspect ratio.</param>
6      /// <param name="zNear">The near depth clipping plane.</param>
7      /// <param name="zFar">The far depth clipping plane.</param>
8      /// <returns>A <see cref="mat4"/> that contains the projection matrix for th
e perspective transformation.</returns>
9      public static mat4 perspective(float fovy, float aspect, float zNear, float
zFar)
10     {
11         var result = mat4.identity();
12         float tangent = (float)Math.Tan(fovy / 2.0f);
13         float height = zNear * tangent;

```

0

```

14         float width = height * aspect;
15         float l = -width, r = width, b = -height, t = height, n = zNear, f = zFar;
16         result[0, 0] = 2.0f * n / (r - l); // = 2.0f * zNear / (2.0f * zNear * tangent * aspect)
17         result[1, 1] = 2.0f * n / (t - b); // = 2.0f * zNear / (2.0f * zNear * tangent)
18         //result[2, 0] = (r + l) / (r - l); // = 0.0f
19         //result[2, 1] = (t + b) / (t - b); // = 0.0f
20         result[2, 2] = -(f + n) / (f - n);
21         result[2, 3] = -1.0f;
22         result[3, 2] = -(2.0f * f * n) / (f - n);
23         result[3, 3] = 0.0f;
24
25         return result;
26     }

```

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perspective用怎样的参数计算得到的。（当然，并非所有矩阵都能用perspective计算出来）

```

1      /// <summary>
2      /// 如果此矩阵是glm.perspective()的结果，那么返回glm.perspective()的各个参数值。
3      /// </summary>
4      /// <param name="matrix"></param>
5      /// <param name="fovy"></param>
6      /// <param name="aspectRatio"></param>
7      /// <param name="zNear"></param>
8      /// <param name="zFar"></param>
9      /// <returns></returns>
10     public static bool TryParse(this mat4 matrix,
11         out float fovy, out float aspectRatio, out float zNear, out float zFar)
12     {
13         float tanHalfFovy = 1.0f / matrix[1, 1];
14         fovy = 2 * (float)(Math.Atan(tanHalfFovy));
15         if (fovy < 0) { fovy = -fovy; }
16         //aspectRatio = 1.0f / matrix[0, 0] / tanHalfFovy;
17         aspectRatio = matrix[1, 1] / matrix[0, 0];
18         if (matrix[2, 2] == 1.0f)
19         {
20             zFar = 0.0f;
21             zNear = 0.0f;
22         }
23         else if (matrix[2, 2] == -1.0f)
24         {
25             zNear = 0.0f;
26             zFar = float.PositiveInfinity;
27         }
28         else
29         {
30             zNear = matrix[3, 2] / (matrix[2, 2] - 1);
31             zFar = matrix[3, 2] / (matrix[2, 2] + 1);
32         }
33         if (matrix[0, 0] == 0.0f || matrix[1, 1] == 0.0f || matrix[2, 2] == 0.0f)
34             return false;
35     }

```

```
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\(go to top\)](#)

编辑

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](#)

好文要顶

关注我

收藏该文



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

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) 网站首页。

0

【推荐】超50万C++/C#源码：大型实时仿真组态图形源码
【推荐】零基础轻松玩转云上产品，获赠礼加返百元大礼
【推荐】华为IoT平台开发者套餐9.9元起，购买即送免费课程



Copyright © 2019 BIT祝威
Powered by .NET Core 3.0 Preview 8 on Linux

Large Visitor Globe

