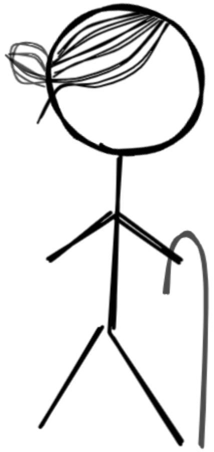
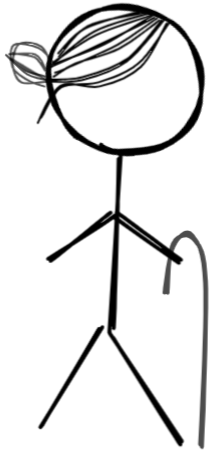
Lecture 2:



图形与渲染技术

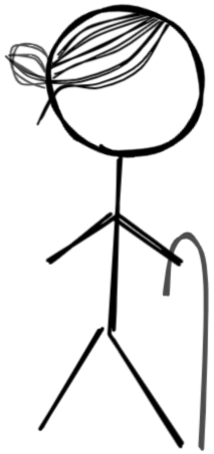
-- 孙绍彬 71184501138@stu.ecnu.edu.cn

Lecture 2:



图形流水线

Lecture 2:

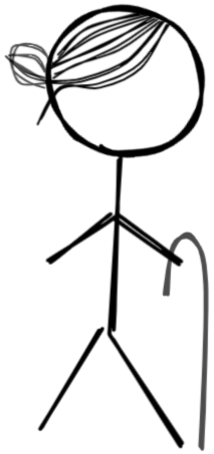


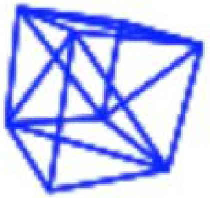
问题1:能够使用canvas进行3D渲染吗？

问题2:能够使用WebGL进行2D渲染吗?

问题3:2D渲染和3D渲染到底是怎么定义的?

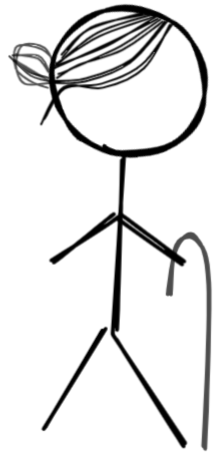
Lecture 2:





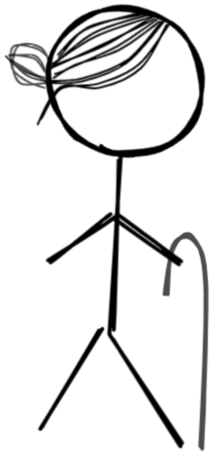
图一

Lecture 2:

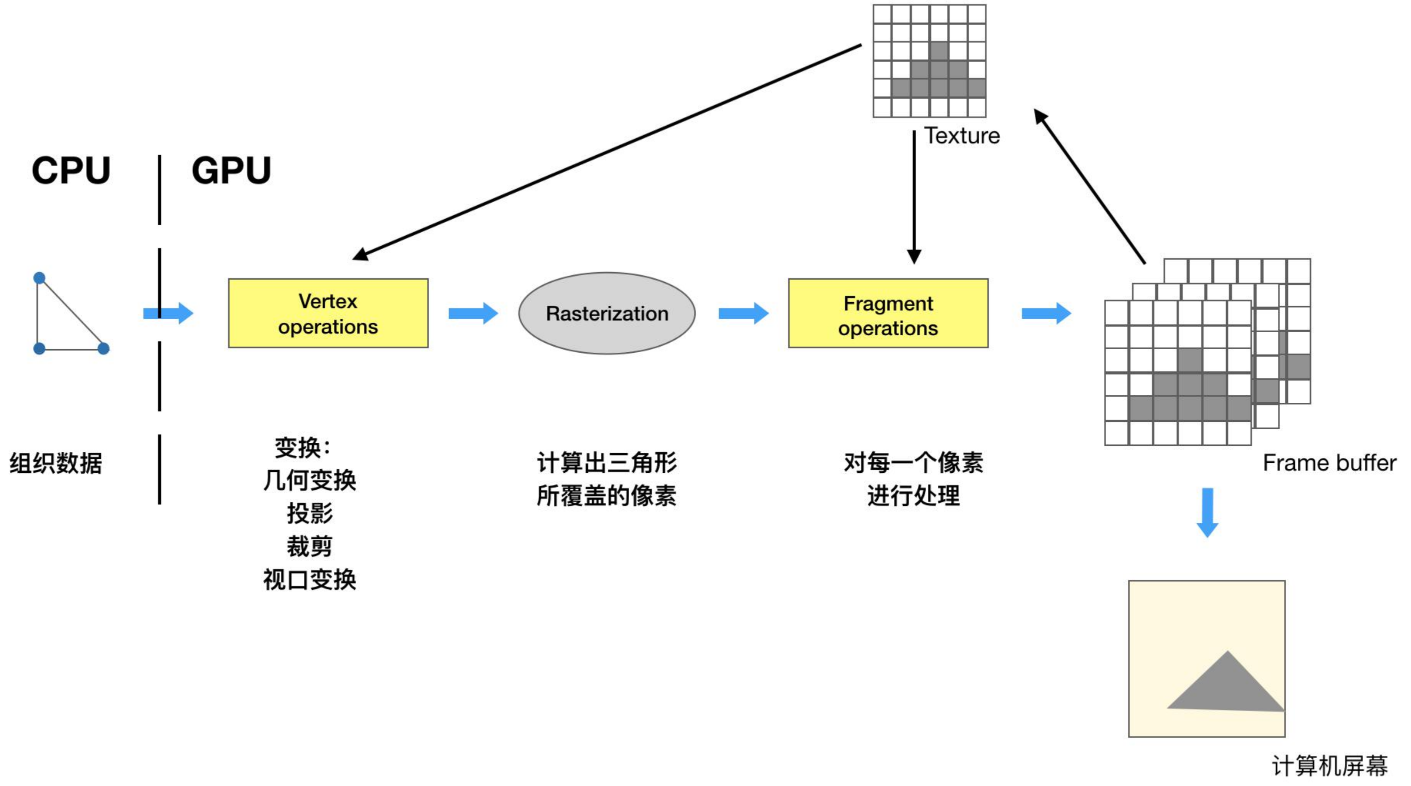


1.可使用canvas渲染3D场景
2.可以使用WebGL渲染(是否有必要)
3.开放讨论

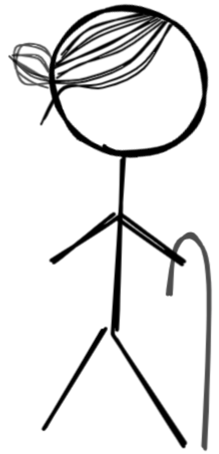
Lecture 2:



图形流水线

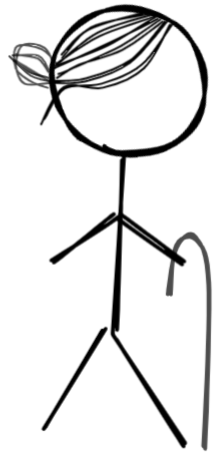


Lecture 2:



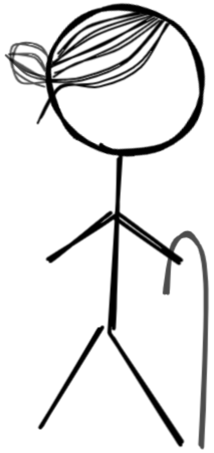
2.GPU与CPU

Lecture 2:



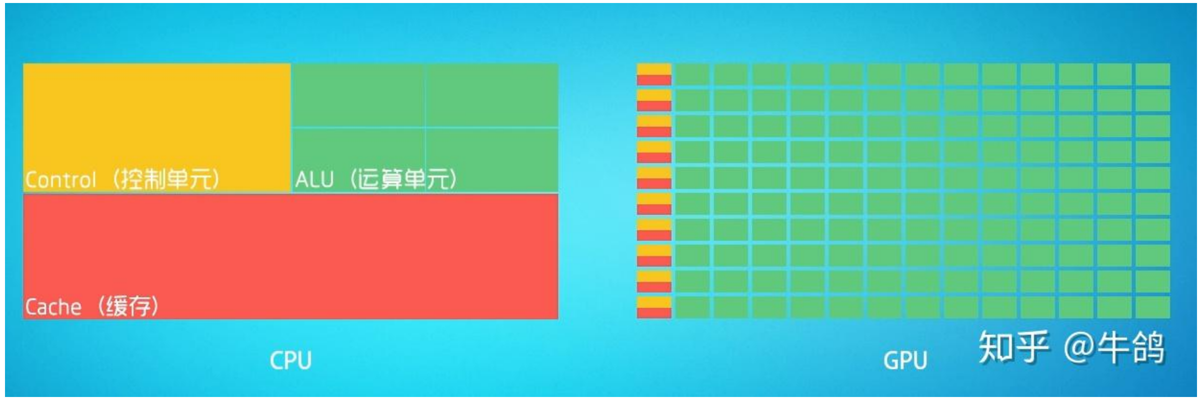
CPU计算特点与GPU?

Lecture 2:



CPU：可以形象的理解为有25%的ALU(运算单元)、有25%的Control(控制单元)、50%的Cache(缓存单元)

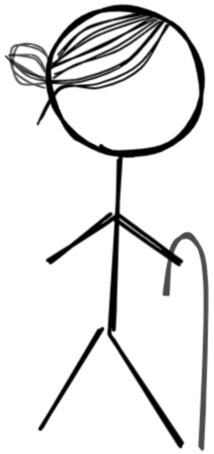
特点：需要少量的运算单元，强大的逻辑运算能力，可以理解为4个专家，既可以做奥数题，也可以做加减法
需要足够的控制单元实现复杂的数据控制和数据转发
需要足够的缓存单元去存放一些已经计算完成的结果，或者是后面马上要用到的数据



GPU：可以形象的理解为90%的ALU(运算单元)，5%的Control(控制单元)、5%的Cache(缓存单元)

特点：大量的运算单元：负责简单粗暴的计算，不擅长奥数题，但小学题他会
少量的控制单元和缓存单元：主要是负责合并和转发数据，对这两块的需求较小，所以占据GPU较小的空间

Lecture 2:

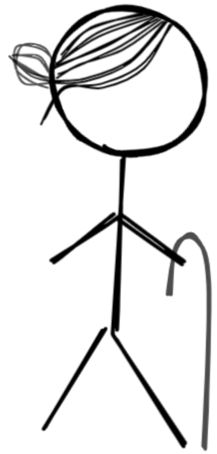


对于多个1+1算术题的计算速度比较

CPU：速度较慢。因为计算原理是：先算第1题， 再算第2题，总时间为【T1+T2+T3>>>+T1000（也就是1000个算术题消耗时间的累加]）】

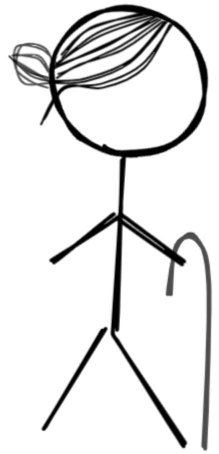
GPU：速度很快。因为计算原理是：可同时计算1000道算术题，总时间为【max（T1,T2,T3...T1000）（也就是1000个算术题消耗时间中的最大值）】

Lecture 2:



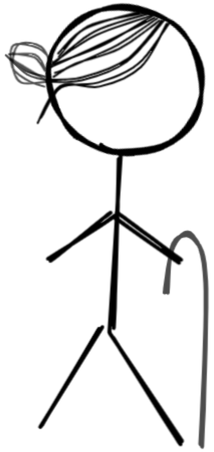
浏览器中canvas的计算?

Lecture 2:



Canvas（JavaScript语句）
JavaScript引擎
浏览器接口逻辑
图形库

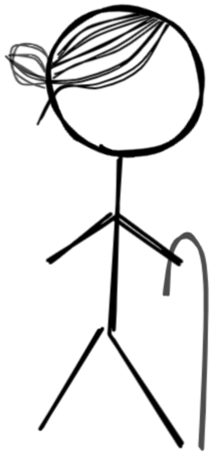
Lecture 2:



利用GPU通用计算

https://zhuanlan.zhihu.com/p/38992506
https://github.com/gpujs/gpu.js

Lecture 2:

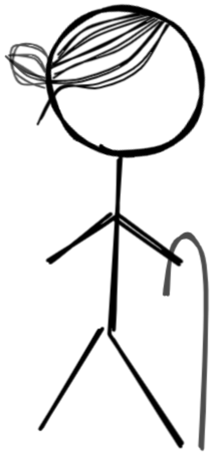


顶点着色器

```
var VSHADER_SOURCE =
`attribute vec4 a_Position;
void main() {
        gl_Position = a_Position;
}`;
```

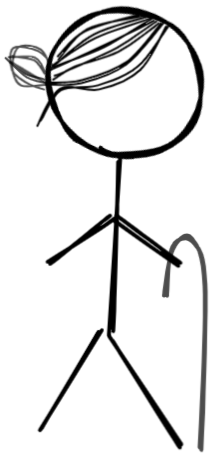
Lecture 2:

片元着色器



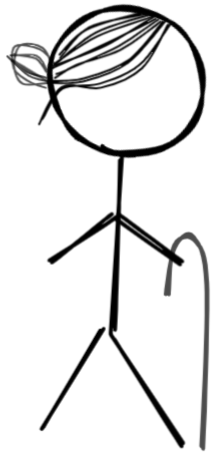
```
var FSHADER_SOURCE =
`precision mediump float;
uniform float u_Width;
uniform float u_Height;
void main() {
    //证明片元着色器是逐像素被调用的
        gl_FragColor = vec4(gl_FragCoord.x /
u_Width, 0.0, gl_FragCoord.y / u_Height, 1.0);
}`;
```

Lecture 2:



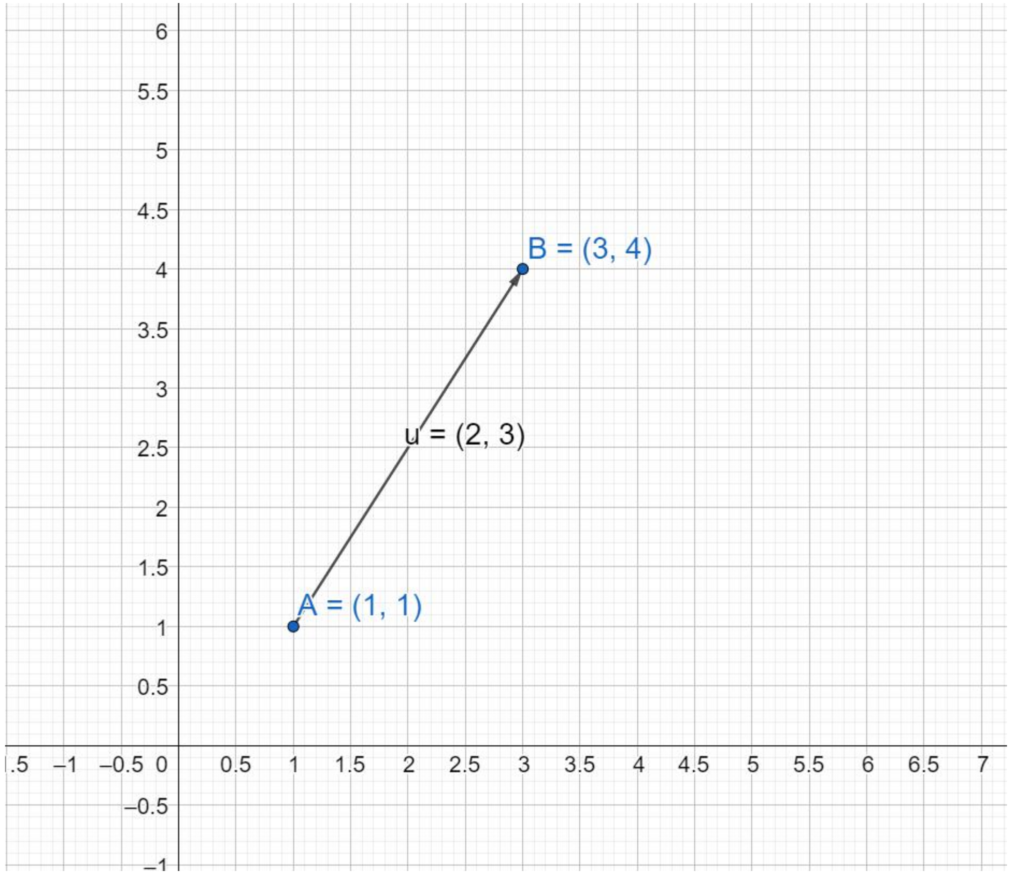
矢量

Lecture 2:

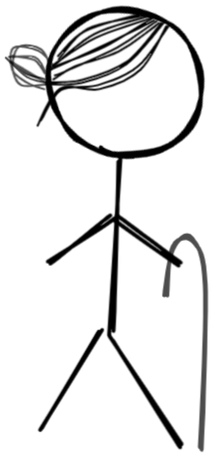


$$\mathbf{u} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

对于有向线段 u 来说，它的向量表示保存了两条信息：方向和长度。
而对于坐标点 A 和 B 来说，实际上也隐含了两条类似的信息：原点到坐标点的方向和原点到坐标点的距离长度。方向通过向量各分量的比例而确定，而长度则是通过向量大小（magnitude）确定。



Lecture 2:



$$|u| = \sqrt{x^2 + y^2}$$

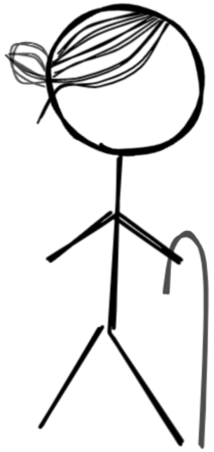
向量大小
向量大小写作 $|u|$(通过勾股定理)，其中 x 和 y 表示 u 的两个分量。

$$\hat{u} = \frac{u}{|u|}$$

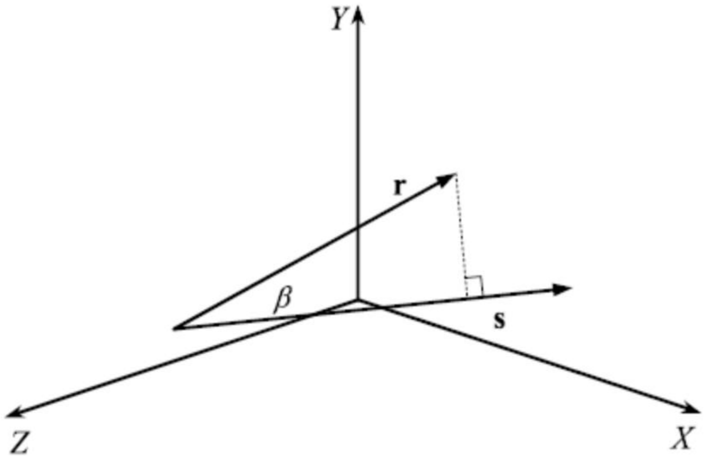
单位向量就是大小为 1 的向量，把普通向量转换为单位向量的过程称为规范化或标准化（normalization）。

向量的规范化很容易，将向量除以它的大小即可。

Lecture 2:



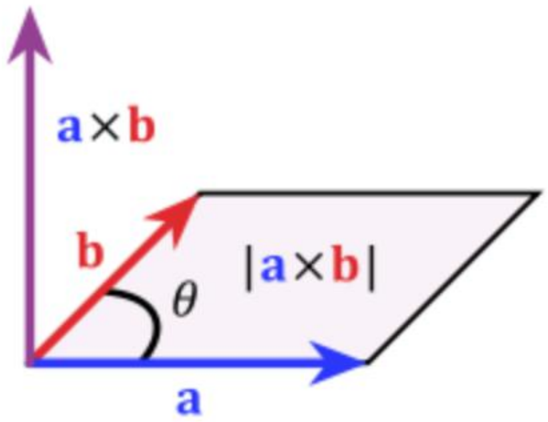
$$r \cdot s = |r|\,|s|\,cos\beta$$



几何意义：向量 r 对 s 作投影，得到 $rcos\beta$，再将投影的大小 $|r|cos\beta$ 和 s 向量的大小 $|s|$ 相乘

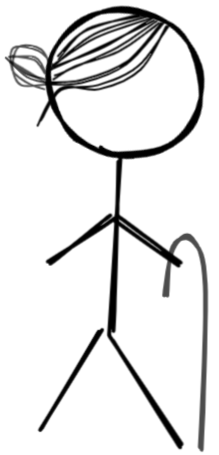
意义：点乘在计算光照的时候非常有用，比如在聚光灯的效果计算中，可以根据点乘来得到光照效果，如果点乘越大，说明夹角越小，则物体离光照的轴线越近，光照越强。在 GLSL 中内置了点乘函数 dot(v1, v2)。

$$|t| = |a|\,|b|\,sin\theta$$



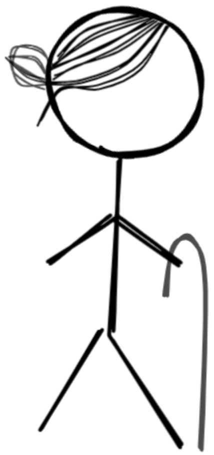
大小等于两个向量张成的平行四边形的面积。该面积衡量了两个向量的差异性。如果 a 和 b 是垂直（正交）的，两者的差异性最大，在两个向量大小不变的情况此时面积最大（$sin\theta$=1）；如果 a 和 b 是共线的，两者的差异性最小，此时面积等于 0

Lecture 2:



矩阵

Lecture 2:



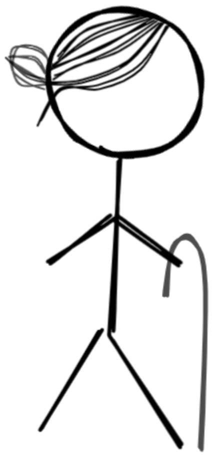
矩阵是按照行列排列的一系列数值的集合，一个矩阵通常是由 m 行 n 列组成，我们称之为m x n 矩阵。在 GLSL 着色器语言中我们可以使用 mat2、mat3、mat2x3、mat3x4 等来表示不同行列数的矩阵，如 mat3 表示行列数为 3x3 的矩阵。

$$M = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

矩阵与矩阵相乘，需要满足左侧矩阵的列数与右侧矩阵的行数相等。结果矩阵的行数为左侧矩阵的行数，列数等于右侧矩阵的列数。

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\cdot5+2\cdot7 & 1\cdot6+2\cdot8 \\ 3\cdot5+4\cdot7 & 3\cdot6+4\cdot8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Lecture 2:

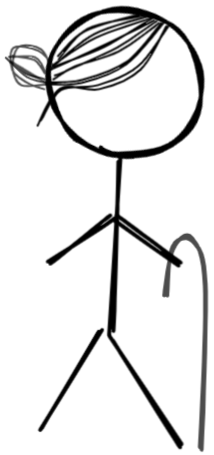


设某点向x方向移动 dx, y方向移动 dy ,[x,y]为变换前坐标， [X,Y]为变换后坐标。
则 X = x+dx; Y = y+dy;
以矩阵表示：

$$[X, Y, 1] = [x, y, 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix};$$

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{matrix}$$ 即平移变换矩阵。

Lecture 2:



设某点坐标，在x轴方向扩大 sx倍，y轴方向扩大 sy倍，[x,y]为变换前坐标， [X,Y]为变换后坐标。

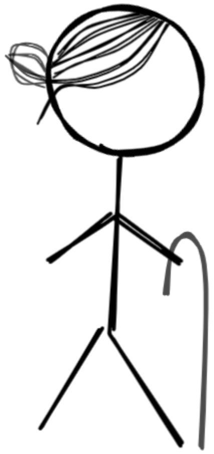
$X = sx*x; Y = sy*y;$

则用矩阵表示：

$$[X, Y, 1] = [x, y, 1]\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$\begin{matrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{matrix}$$ 即为缩放矩阵。

Lecture 2:



设某点与原点连线和X轴夹角为b度，以原点为圆心，逆时针转过a度 ，原点与该点连线长度为R, [x,y]为变换前坐标， [X,Y]为变换后坐标。

x = Rcos(b) ; y = Rsin(b);

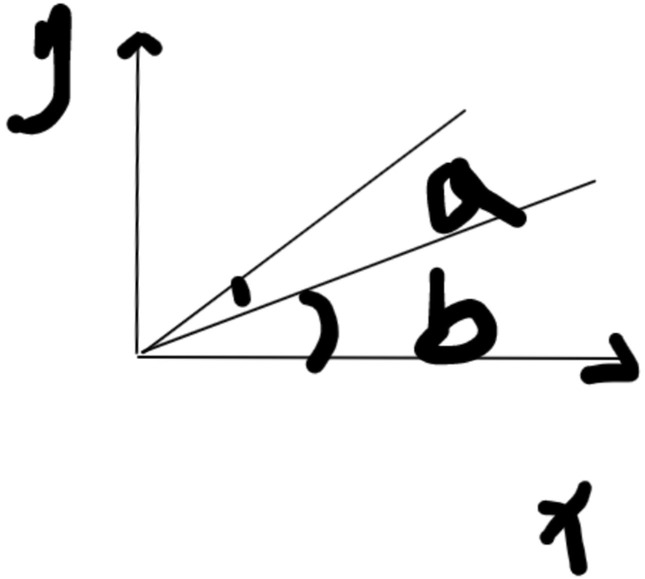
X = Rcos(a+b) = Rcosacosb - Rsinasinb = xcosa - ysina;

Y = Rsin(a+b) = Rsinacosb + Rcosasinb = xsina + ycosa ;

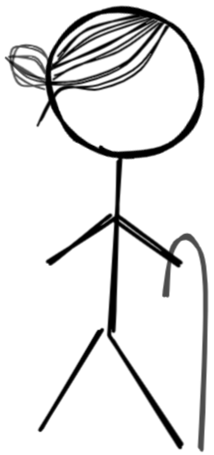
用矩阵表示：

$$[X, Y, 1] = [x, y, 1]\begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{matrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{matrix}$$ 为旋转变换矩阵。（顺时针旋转）

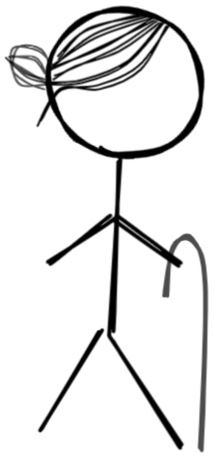


Lecture 2:



作业1：
用原生WebGL实现一个长方形的渲染

Lecture 2:



Thanks!