Transformation

- **Homogeneous Coordinates Representation（齐次坐标表示）**
  - Point(x,y,z,h),
  - Vector(x,y,z,0)
- **Affine Transformation(仿射变换):**
  - Standard Transformation：Translation, Rotation, Scaling, Reflection, Shear.
  - Use Homogeneous Coordinates, Transformation is a 4*4 Matrix, 12 freedom
  - Use column vector representation , affine transformation then like:

$$M_{4*4} = \begin{bmatrix} sx & b & c & tx \\ d & sy & f & ty \\ g & h & sz & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Concatenation Transformation（串联变换）:**

p’ = Mn*…*M2*M1* p; //Mi is Standard Transformation

M=Mn*…*M2*M1; //M is Concatenation Transformation

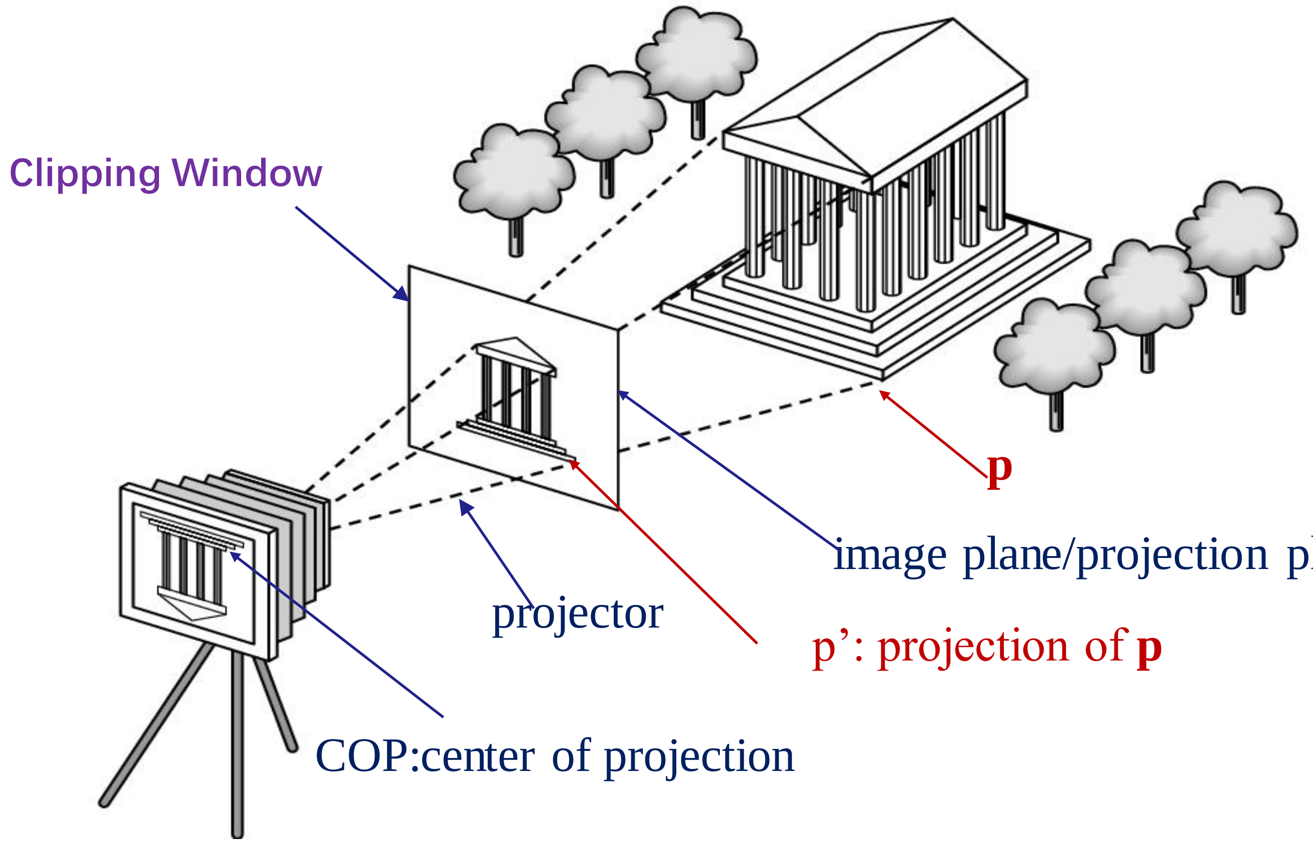
Outline

- Viewing Process 观察流程中的串联变换

- Model-View Transformation（模视变换）
  - Model Transformation模型变换*
  - View Transformation观察变换*
  - Projection Transformation投影变换
  - Viewport Transformation屏幕映射

Computer Viewing Process

- Synthetic Camera Model虚拟相机模型
  - 从P到P’,顶点坐标如何变换?



The University of New Mexico

Computer Viewing Process（cont.)

➢类比于相机拍摄过程，经历“五”种坐标系和“四”种变换！

物体模型

▪MC模型坐标

▪将物体建模坐标描述转换到世界坐标场景下描述-模型变换-

▪WC世界坐标

▪将物体世界坐标场景描述转换为观察坐标系下描述-观察变换/相机变换

▪VC观察坐标

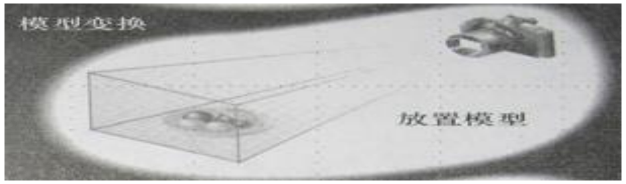
▪将观察坐标裁剪窗口内场景并进行投影变换并映射到规范化坐标下规范化区域-规范化投影变换-

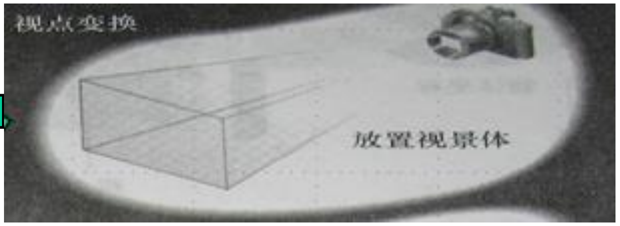
▪NDC规范化坐标

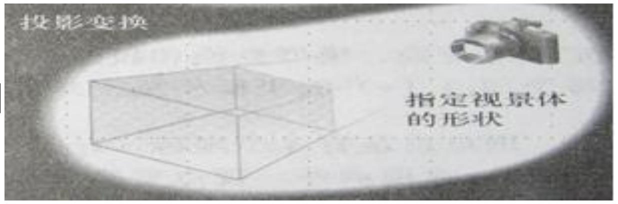
▪将规范化区域景物映射到设备坐标系下视区内-屏幕映射/视口变换

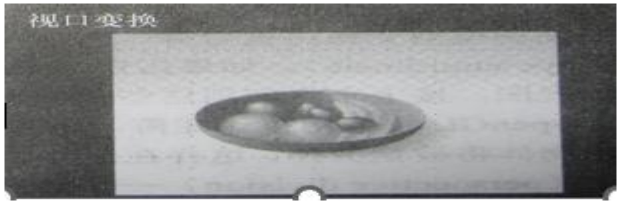
▪DC设备坐标

▪输出设备





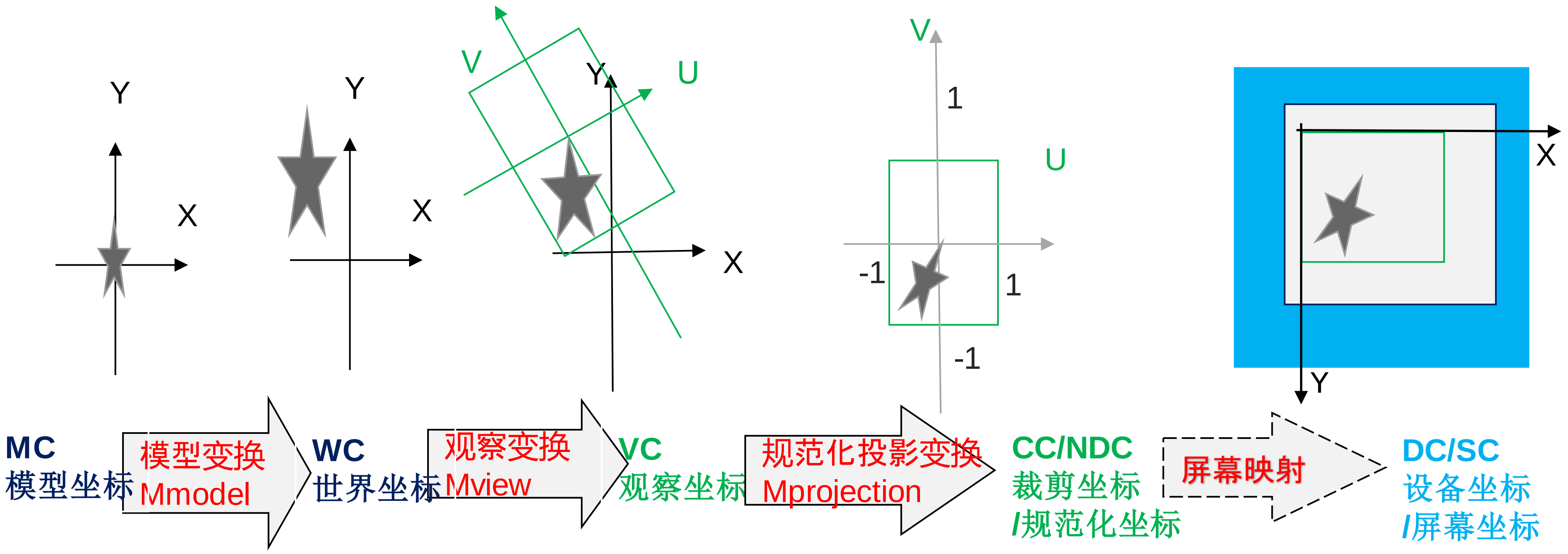




The University of New Mexico

Computer Viewing Process（cont.)

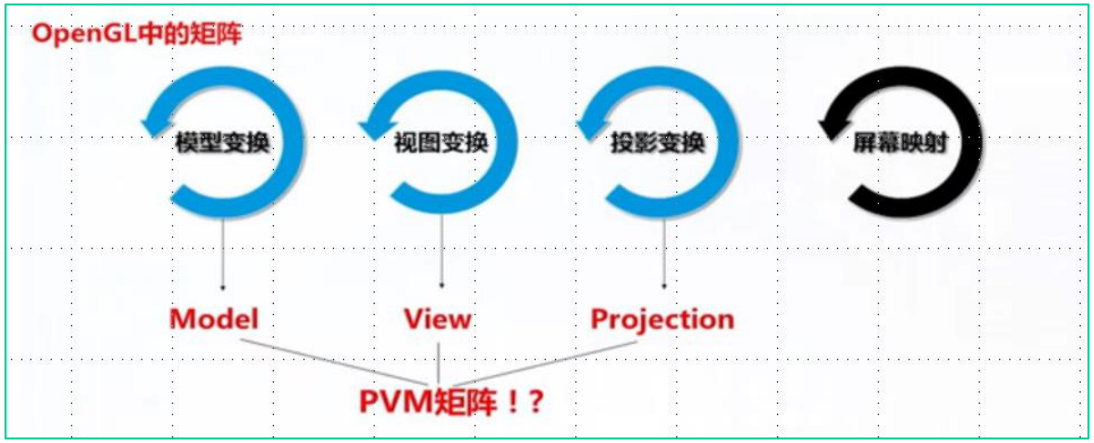
- **Every change of coordinates is equivalent to a concentrate matrix transformation**



- **注：View coordinates观察坐标系：即Camera (Eye) coordinates相机(眼)坐标系**

Computer Viewing Process（cont.)

- ➢Much of the work is in converting object representations from one coordinate system to another（主要工作就是将物体表示，从一个坐标系转换到另一坐标系）



- ●Vertex Shader 顶点着色器中，需要将前三个变换完成，输出裁剪坐标（即NDC坐标）：//简记为MVP矩阵或PVM矩阵

  PVM ＝ Projection Matrix * View Matrix* Model Matrix

  注意1： 以列向量方式表示顶点和变换，所以是左乘顺序

  注意2： Mview * Mmodel 常常合并称为 ModelView矩阵

- “屏幕映射”由系统根据程序设置参数自动完成

设置

The University of New Mexico

1. 观察流水中的坐标变换（级联变换）包含哪些？

A 模型变换

B 相机变换/观察变换

C 规范化投影变换

D 屏幕映射

提交

Outline

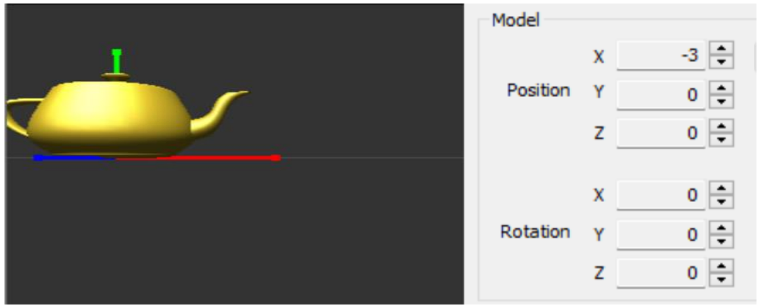
- Viewing Process 观察流程中的级联变换

- **ModelView Transformation**
  - **Model Transformation模型变换**
  - **View Transformation观察变换***

- Projection Transformation
- Viewport Transformation

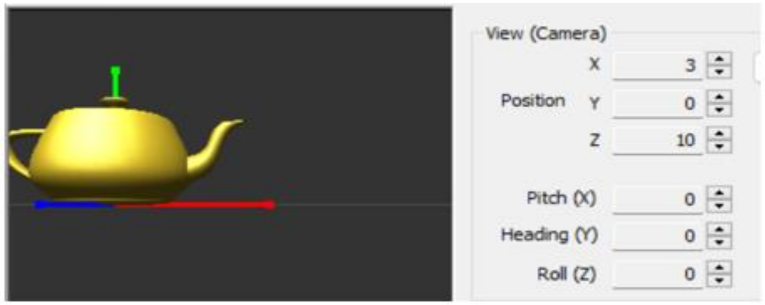
ModelView Transformation

- 模视变换
  - “模视变换”由“模型变换 ”和“观察变换”构成
    - 模型变换Model Transformation：摆放物体
    - 视点变换View Transformation：摆放相机， 也称为”观察变换”或“相机变换”
  - 模型变换和观察变换具有对偶性
    - 同一画面，既可以是模型变换后结果，也可以是相机变换后的结果！



Model

Position	X	-3
	Y	0
	Z	0
Rotation	X	0
	Y	0
	Z	0



View (Camera)

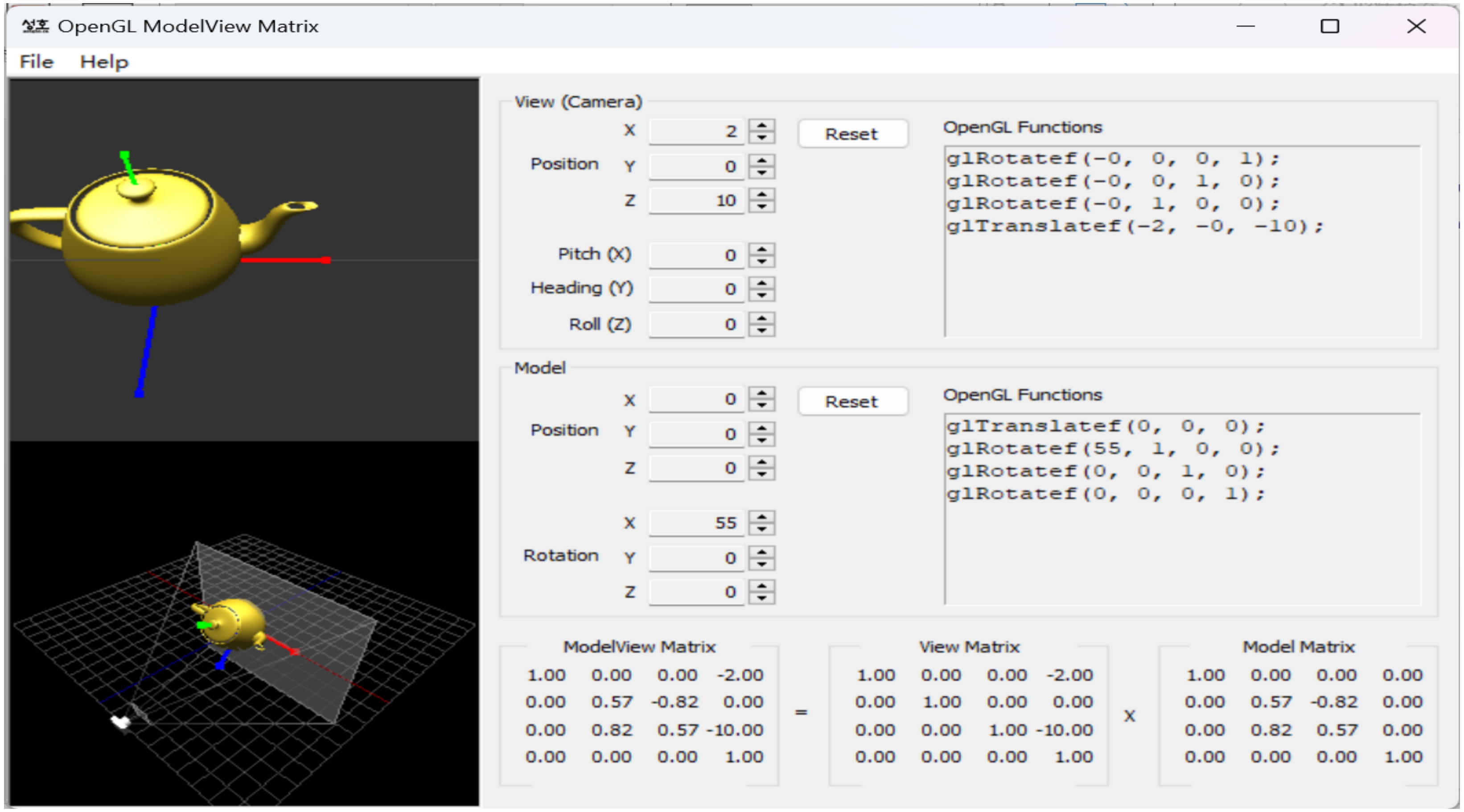
Position	X	3
	Y	0
	Z	10
Pitch (X)		0
Heading (Y)		0
Roll (Z)		0

ModelView Transformation

The University of New Mexico

➢模视变换矩阵

- ModeView Matrix = View Matrix * Model Matrix



View (Camera)

Position	X	2
	Y	0
	Z	10
Pitch (X)		0
Heading (Y)		0
Roll (Z)		0

OpenGL Functions

```
glRotatef(-0, 0, 0, 1);
glRotatef(-0, 0, 1, 0);
glRotatef(-0, 1, 0, 0);
glTranslatef(-2, -0, -10);
```

Model

Position	X	0
	Y	0
	Z	0
Rotation	X	55
	Y	0
	Z	0

OpenGL Functions

```
glTranslatef(0, 0, 0);
glRotatef(55, 1, 0, 0);
glRotatef(0, 0, 1, 0);
glRotatef(0, 0, 0, 1);
```

$$\underset{\text{ModelView Matrix}}{\begin{pmatrix} 1.00 & 0.00 & 0.00 & -2.00 \\ 0.00 & 0.57 & -0.82 & 0.00 \\ 0.00 & 0.82 & 0.57 & -10.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix}} = \underset{\text{View Matrix}}{\begin{pmatrix} 1.00 & 0.00 & 0.00 & -2.00 \\ 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & -10.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix}} \times \underset{\text{Model Matrix}}{\begin{pmatrix} 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.57 & -0.82 & 0.00 \\ 0.00 & 0.82 & 0.57 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix}}$$

Outline

- Viewing Process 观察流程的级联变换

- Model-View Transformation
  - Model Transformation模型变换*
  - View Transformation观察变换*

- Projection Transformation
- Viewport Transformation

Model Transformation

➢Model Transformation模型变换

- 将模型局部坐标转换为场景世界坐标(from MC to WC)
- 3d模型在建立之初，有它自身的坐标系，以及相应的点坐标。如果把它放入一个复杂的3d模型组成的场景中的某个位置，需要将它自身的坐标点全部移动到改位置，这个转换过程就叫 "模型转换"。而3d场景的坐标系，叫做"世界坐标系"。

这个转换过程有以下几种情况发生：

1）坐标系重合：

这种情况是"模型变换"中最简单的情况，不用考虑坐标系的转换过程，只用考虑模型的平移即可.

2) 坐标系不重合

这个变换较为复杂，需要做 旋转，移动等复合变换。

方法为 ：1.将模型坐标系变为与世界坐标系一致的坐标系统 2.在其上执行移动操作.

Model Transformation(cont.)

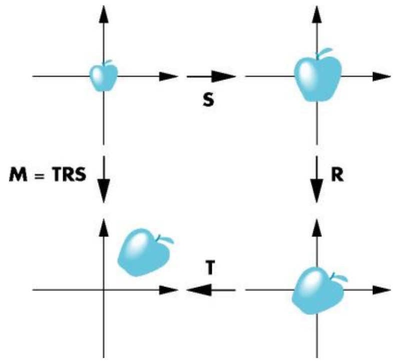
- Instance Transformation实例变换
  - a prototype object ~a symbol~图符(原型)
  - Each appearance of the object in the model is~ an instance~实例
    - Must scale, orient, position 缩放，定向和定位
  - Defines instance transformation

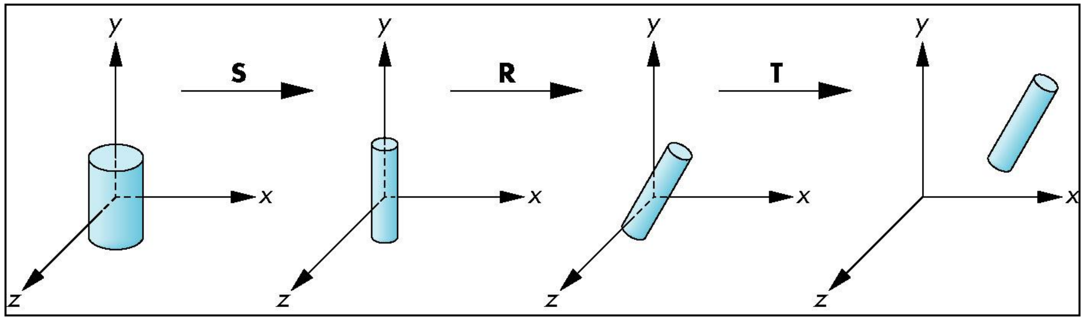
  General: M=TRS

  Scale: S

  Orient:R

  Locate:T





The University of New Mexico

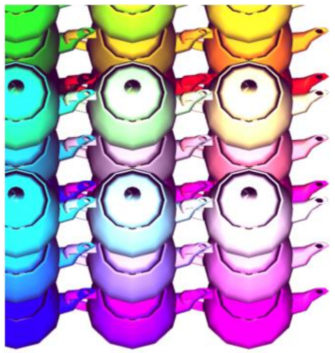
Model Transformation(cont.)

- Model Transformation模型变换
  - Instance Transformation实例变换(cont.)

Example1: WebGL2 性能优化 - 实例化绘制

https://webgl2fundamentals.org/webgl/lessons/zh_cn/webgl-instanced-drawing.html

Example2:AngelCode8E/09/teapotInstance2.html



单选题 1分

设置

The University of New Mexico

1. 实例变换Instance Transformation通常包含以自己为中心的缩放，旋转，以及平移操作，请问一般级联变换应该应该为？(注意：顶点是列向量表示，采用齐次坐标)

A RST

B SRT

C TSR

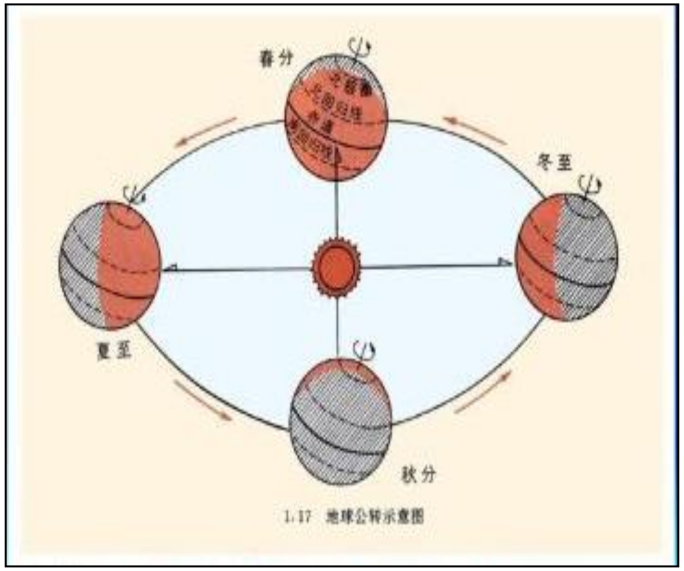
[✓ D] TRS

提交

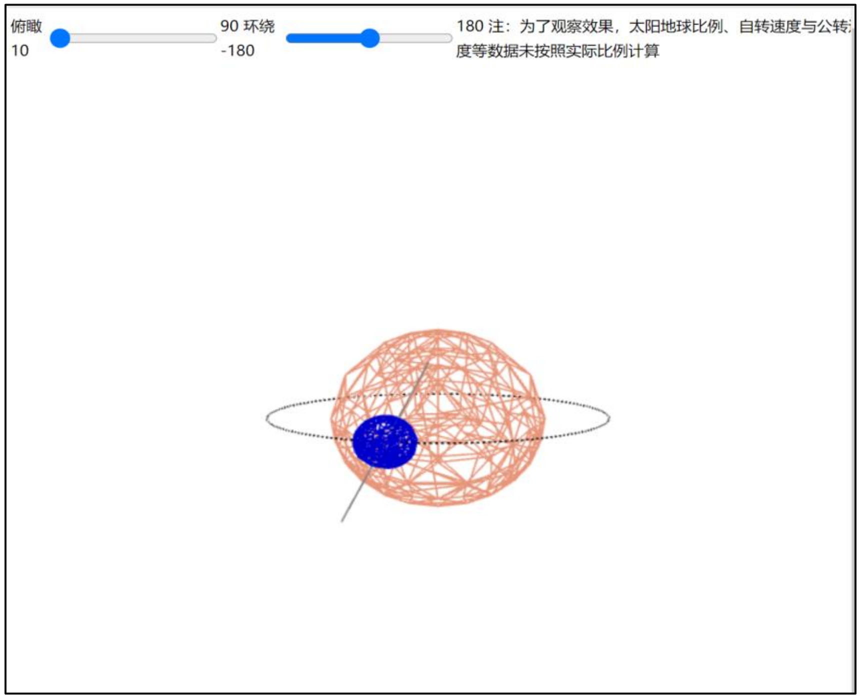
The University of New Mexico

Model Transformation(cont.)

- ➢Complex Model Transformation
  - ➢ Sun-planet Example（演示）



1.17 地球公转示意图


俯瞰 10
90 环绕 -180
180 注：为了观察效果，太阳地球比例、自转速度与公转
度等数据未按照实际比例计算

The University of New Mexico

Model Transformation(cont.)

➢Complex Model Transformation(cont.)

➢ Sun-planet Example（cont.)

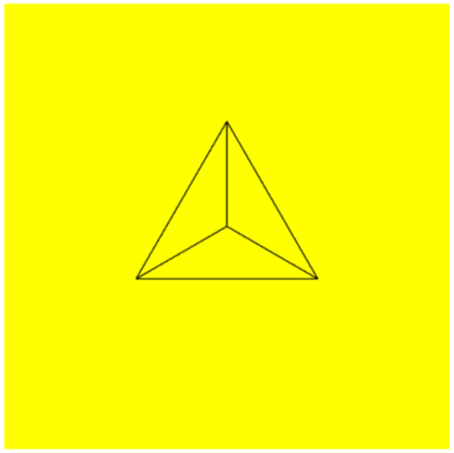
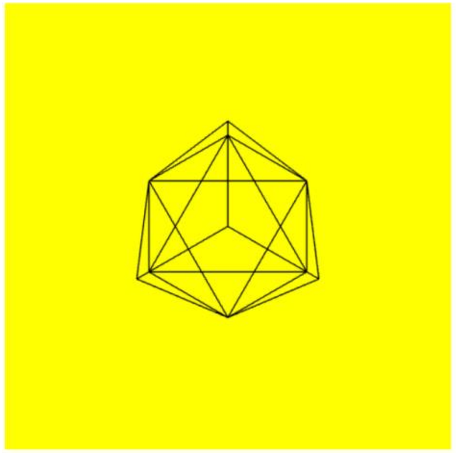
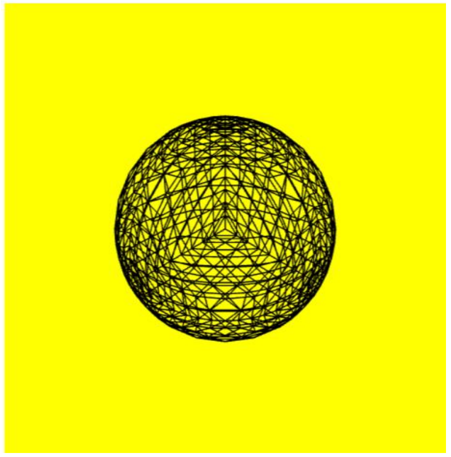
➢球体建模：四面体递归细分得到球面网格mesh

//ref AngleCode/chap5/ Wiresphere.js

step1.初始为在单位四面体，有四个面，四个顶点在单位球面上。

Step2.取每条边上的中点P，将每个面划分为4个面，然后对新顶点进行归一化(长度归约到1，即把新顶点撑到单位球面上）

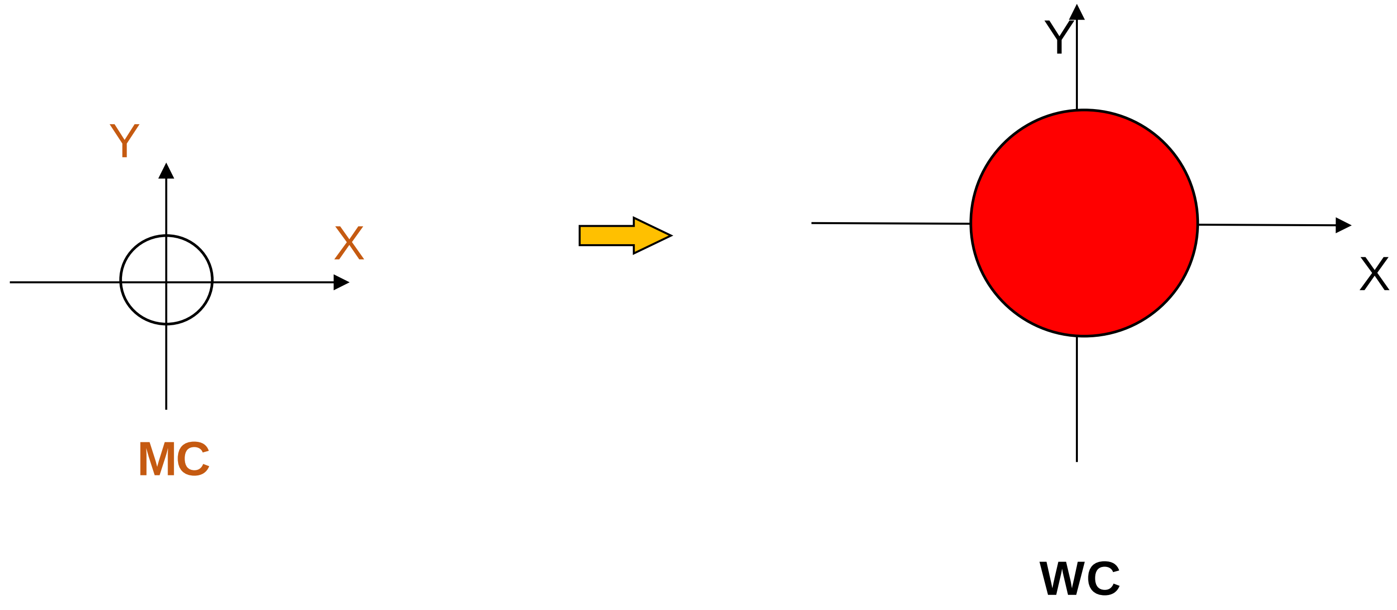
step3.重复step2得到球面网格

The University of New Mexico

Model Transformation(cont.)

- Complex Model Transformation(cont.)
  - Sun-planet Example（cont.)
  - Sun: M_{sun}= Scale(sx,sy,sz)



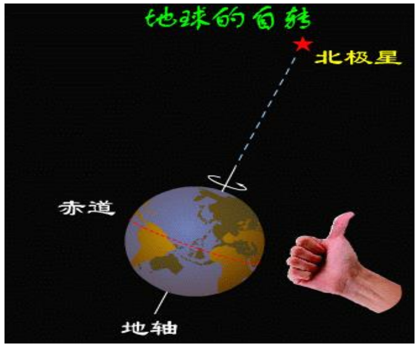
The University of New Mexico

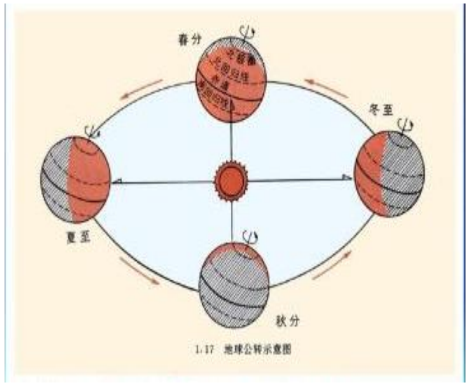
Model Transformation(cont.)

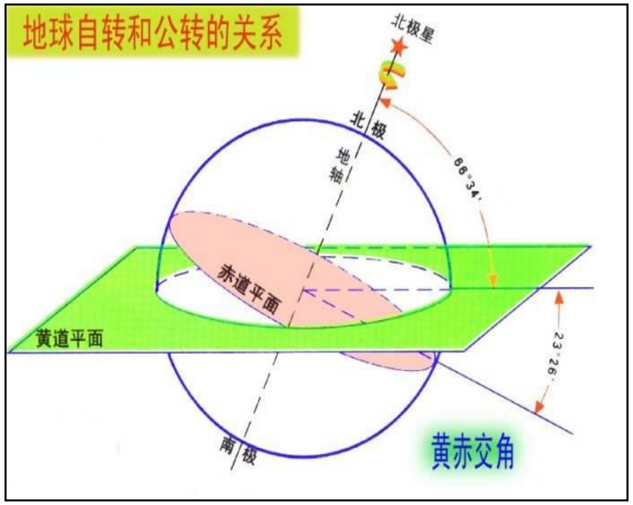
➢Complex Model Transformation(cont.)

➢ Sun-planet Example（cont.)

■**Planet:**　Mplanet= ?







□级联变换矩阵的构成：

1. 球体原型进行缩放到需要的地球尺寸：S（ sx,sy,sz ）
2. 地球绕“地轴”（y轴）逆时针“**自转**”：Ry（**alpha** ）
3. 地球**地轴的偏移**：即绕z轴顺时针旋转Rz(-phi)　*//(Phi = 23.26)*
4. 地球绕太阳的公转的变换矩阵= ？

Earth’=（ ？ ）***** **Rz(-Phi)** ***** **Ry(alpha)** ***** S(sx,sy,sz)* **Earth**

单选题 1分

设置

The University of New Mexico

2.地球绕太阳公转的变换矩阵是哪一个？

（r表示地球距离太阳的距离，theta是地球绕太阳逆时针旋转角度）

[✓ A] T(r*cos(theta), r*sin(theta))

B T(r*cos(-theta), r*sin(-theta))

C R(theta)T(r,0,0)

D R(-theta)T(r,0,0)

提交

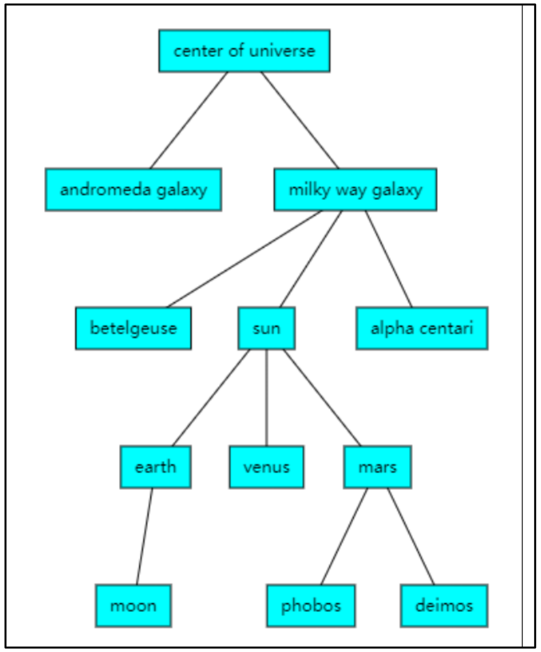
The University of New Mexico

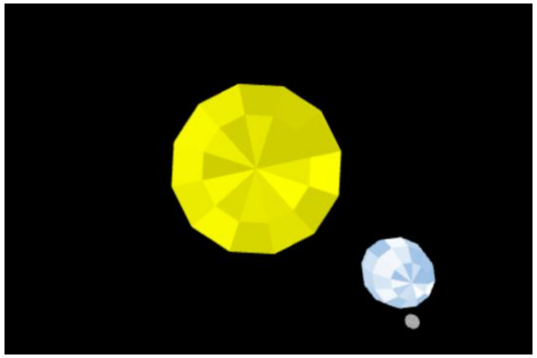
Model Transformation（cont.)

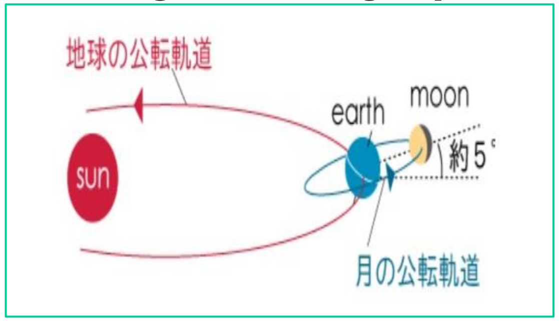
- Complex Model Transformation(cont.)
  - 扩展：场景图scene-graph：具有”层级结构”
    - 大多数三维引擎使用“场景图”，将需要显示的东西放在场景图中，引擎会遍历场景图找出需要绘制的东西。

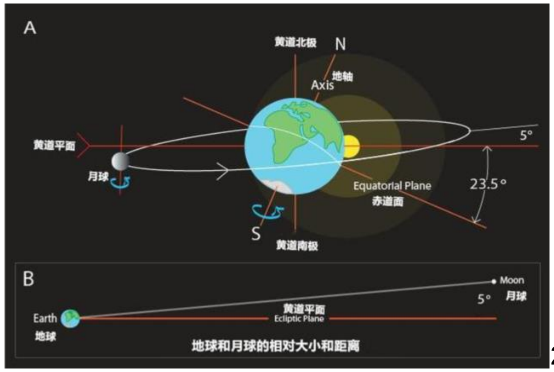
example1：太阳-地球-月亮. (月亮绕着地球转，和地球一起绕着太阳转)

https://webgl2fundamentals.org/webgl/lessons/zh_cn/webgl-scene-graph.html



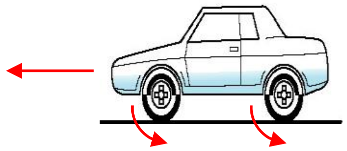




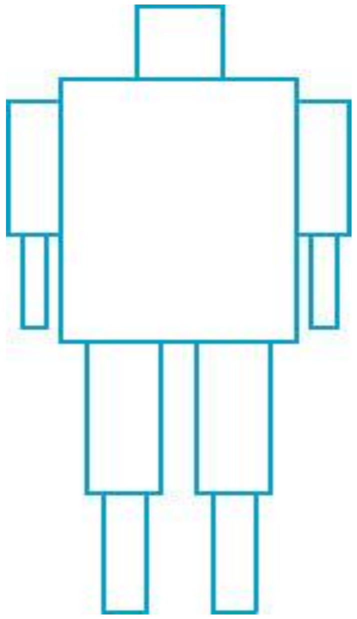


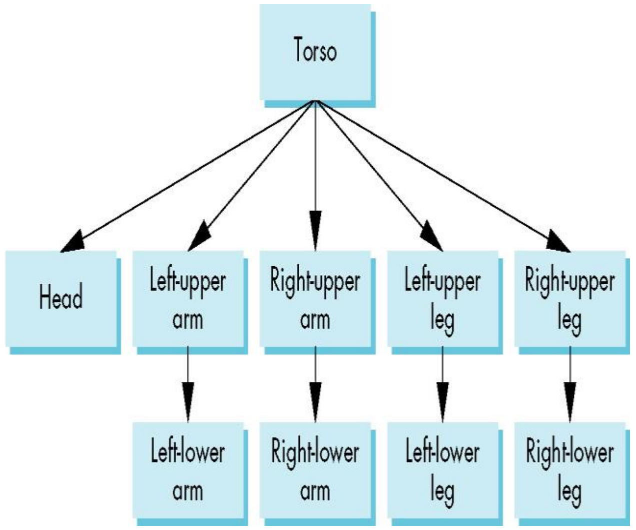
Model Transformation（cont.)

The University of New Mexico



- Complex Model Transformation(cont.)
  - 扩展：场景图scene-graph：具有”层级结构”(继续）
    - Build a tree-structured model of a humanoid figure(类人机器人)（具有铰链结构）
    - *example2　机器人模拟　Ref：Angel8ECode/09/figure.html*





Outline

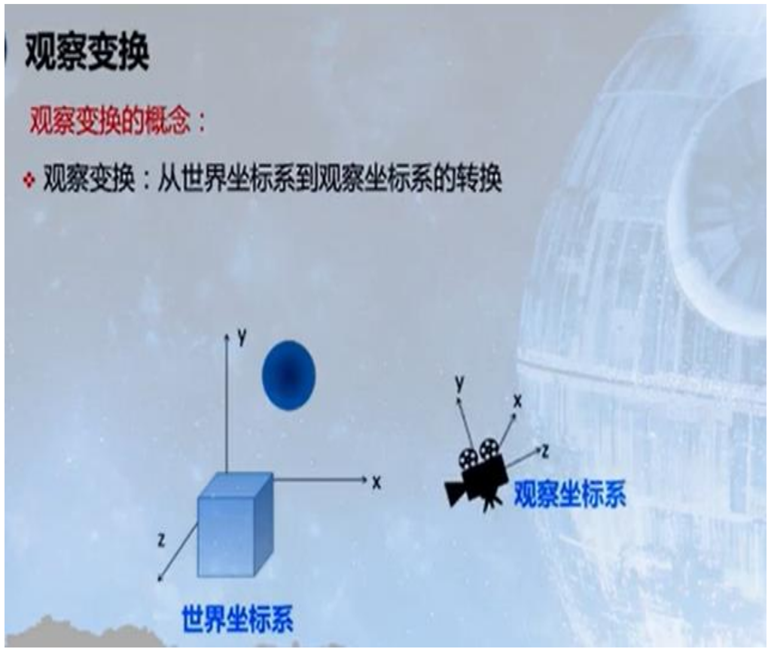
- Viewing Process观察流程的级联变换

- **Model-View Transformation**
  - **Model Transformation模型变换***
  - **View Transformation观察变换***

- Projection Transformation
- Viewport Transformation

View Transformation*

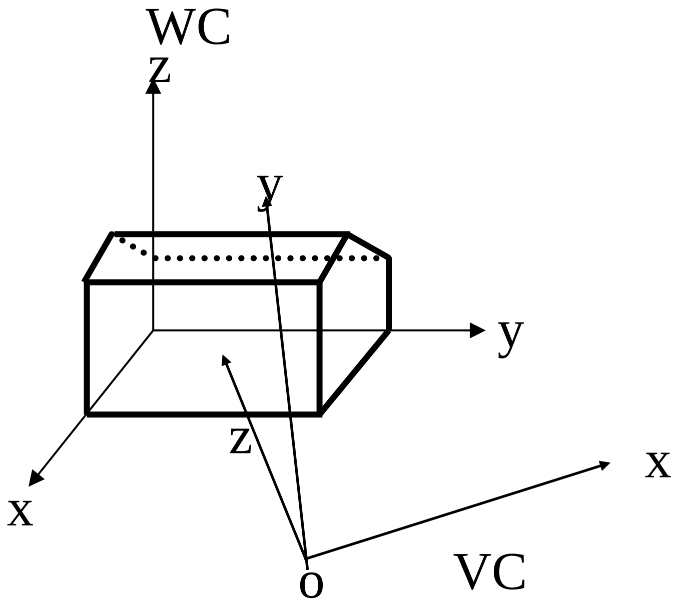
- **View Transformation**
  - 从世界坐标系到观察坐标的变换 (from WC to VC)
  - “观察变换”: 也称 “视点变换”，“相机变换”, “眼坐标变换”



观察变换

观察变换的概念：

- 观察变换：从世界坐标系到观察坐标系的转换



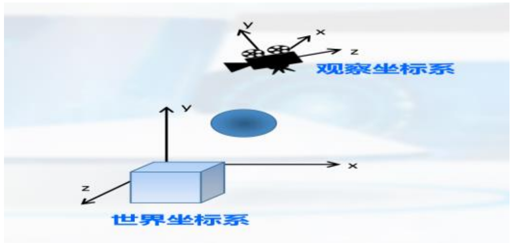
The University of New Mexico

View Transformation（cont.)

- 坐标系变换实质是框架变换（Change of Frames）

Extending what we did with change of bases

$u_1 = \gamma_{11}v_1+\gamma_{12}v_2+\gamma_{13}v_3$
$u_2 = \gamma_{21}v_1+\gamma_{22}v_2+\gamma_{23}v_3$
$u_3 = \gamma_{31}v_1+\gamma_{32}v_2+\gamma_{33}v_3$
$Q_0 = \gamma_{41}v_1+\gamma_{42}v_2+\gamma_{43}v_3 +\gamma_{44}P_0$



Within the two frames, any point or vector has a representation of the same form
$\mathbf{a}=[\alpha_1\ \alpha_2\ \ \alpha_3\ \alpha_4]$ in the first frame(u1,u2,u3,Q0)
$\mathbf{b}=[\beta_1\ \beta_2\ \ \beta_3\ \beta_4]$ in the second frame(v1,v2,v3,P0)
where $\alpha_4 = \beta_4 = 1$ for points and $\alpha_4 = \beta_4 = 0$ for vectors .
defining a 4 x 4 matrix

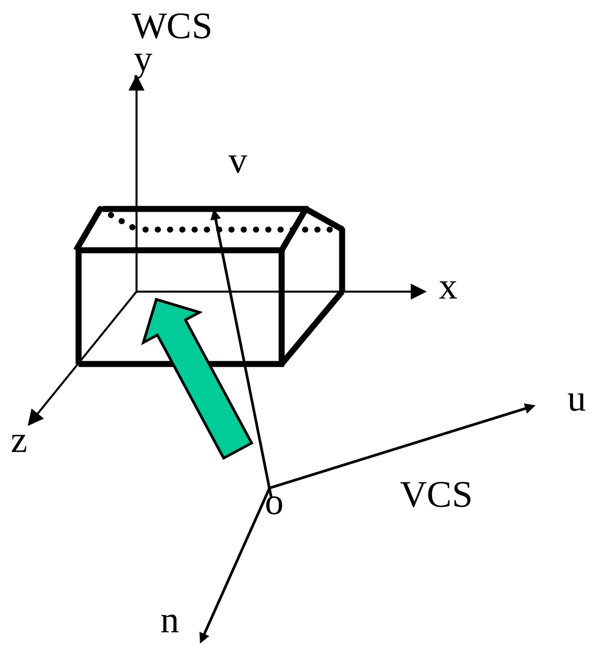
$$\mathbf{M} = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

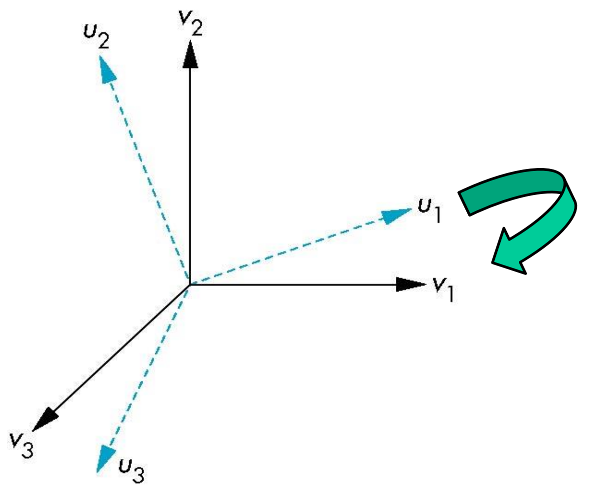
$$\mathbf{a}=\mathbf{M}^T\mathbf{b}$$

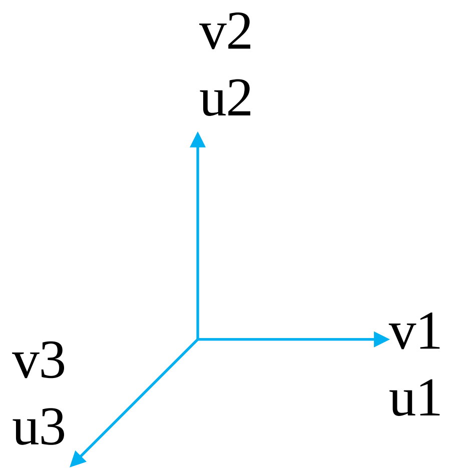
The matrix **M** is 4 x 4 and specifies <u>an affine transformation in homogeneous coordinates</u>

View Transformation(cont.)

- 观察变换(级联变换矩阵)的推导方法
- View Transformation is the Combination transformation that makes the VCS and WCS coincide(重合): **"move VCS to WCS"**，**Commonly like: M= R * T**





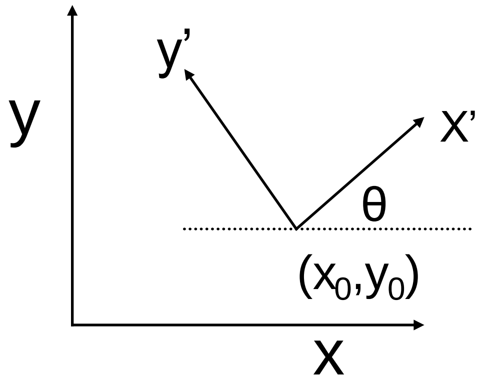


The University of New Mexico

View Transformation(cont.)

➢2D View Transformation

- How to transform the representations of object's locations from XY to X'Y'?
- Example:



✓思路：将 X'O'Y' 变换到与XOY重合

1.平移 $(-x_0,-y_0)$到(0,0)：T$(-x_0,-y_0)$

2.旋转x'y'到xy：R(- θ)

此观察变换矩阵为： **$M=R(-\theta)T(-x_0,-y_0)$**

The University of New Mexico

View Transformation（cont.)

➢2D View Transformation(cont.)

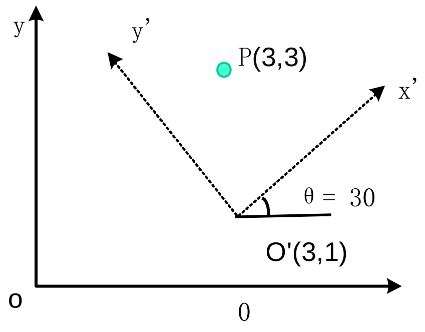
➢Example：O'(3,1),右手坐标系统，Pwc=(3,3),求Mview=?， Pvc =?

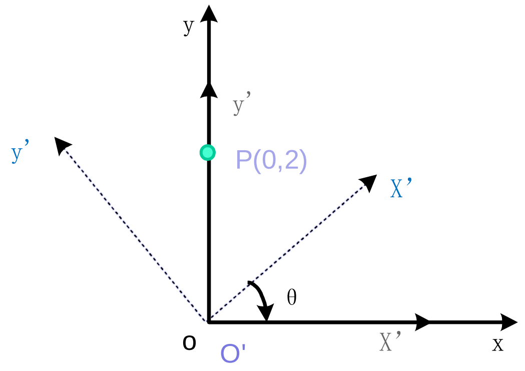
1：将x'y'坐标系的原点平移到xoy坐标系的原点：T(-3,-1)

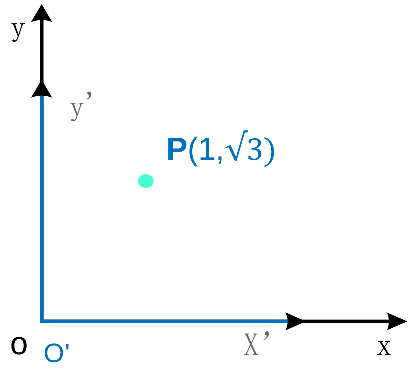
2: 顺时针旋转30度：R(-30)

M_{view}= R(-30) *T(-3,-1)

P_{vc} = R(-30) *T(-3,-1)* P_{wc} =（1，√3）







The University of New Mexico

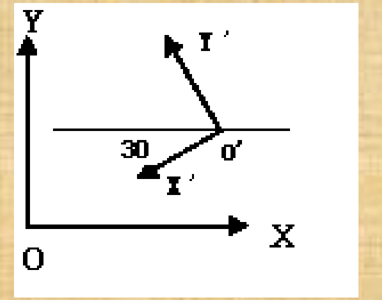
3. 在XOY平面坐标系中，新坐标系的原点O′（8，4，1） 请写出对应的观察变换矩阵？ （注：几何用列向量齐次坐标表示）

A R(30)T(-8,-4)F(Y轴)

B F(Y轴)R(30) T(-8,-4)

[✓ C] F(Y轴)R(-30)T(-8,-4)

[✓ D] R(30) F(Y轴)T(-8,-4)



提交

View Transformation（cont.)

➢ 2D View Transformation（cont.)

？ Mview=R*T，不知道旋转角，R如何求？

- 设R中的第一行为一个向量u，第二行为v,
- 发现u,v正好是VC坐标系的方向轴上的单位正交向量，因：u·v=0，|v|=1，|u|=1 ；
- 所以，旋转变换矩阵R可以直接写为VC坐标系的两个方向轴上的单位向量！

$$R=\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \Rightarrow R=\begin{pmatrix} u_x,u_y \\ v_x,v_y \end{pmatrix}$$

View Transformation（cont.)

➢2D View Transformation(cont.)

Example：已知O’(3,1),右手坐标系统，Pwc=(3,3),求Mview=? ， Pvc =?

➢**计算O’X’上的单位向量u=($\frac{\sqrt{3}}{2}$, 1/2),**

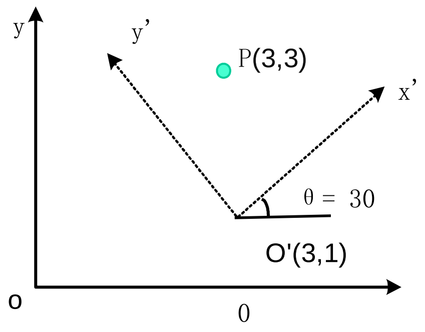
R(u,v)= $\begin{bmatrix} u_x, u_y \\ v_x, v_y \end{bmatrix}$

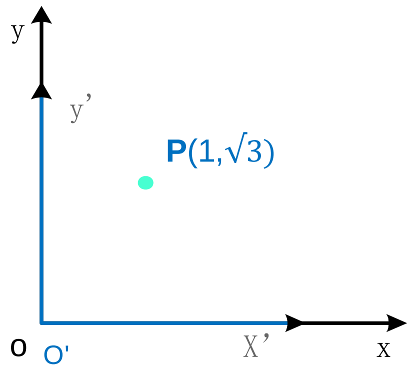
➢**计算O’Y’上的单位向量v=(-1/2, $\frac{\sqrt{3}}{2}$)**

绕原点逆时针转90度得到单位向量v=(v_x,v_y)= (-u_x, u_y)

➢Mview= R(u,v) *T(-3,-1)

➢Pvc = R(u,v) *T(-3,-1)* Pwc = （1，$\sqrt{3}$ ）





The University of New Mexico

View Transformation（cont.)

➢2D View Transformation(cont.)

Practice:新旧坐标都是右手系坐标系统。WC旧坐标系统是XOY, VC新坐标系统是X’O’Y’，O’在XOY坐标系中坐标为（0,0)，点V是O’ Y’轴上的一点，V(3,4)。
请根据单位向量构造旋转矩阵的方法，求从xoy变换到x’o’y’的观察变换矩阵。

➢**计算O’Y’单位向量v: $v=(v_x,v_y)=$ ((Xp-Xo)/r，(Yp-Yo)/r) , $r^2= (Xp-Xo)^2+(Yp-Yo)^2$**

➢**计算O’X’上的单位向量u=v绕原点顺时针转90度得到单位向量u, $u=(u_x,u_y)= (v_y,-v_x)$**

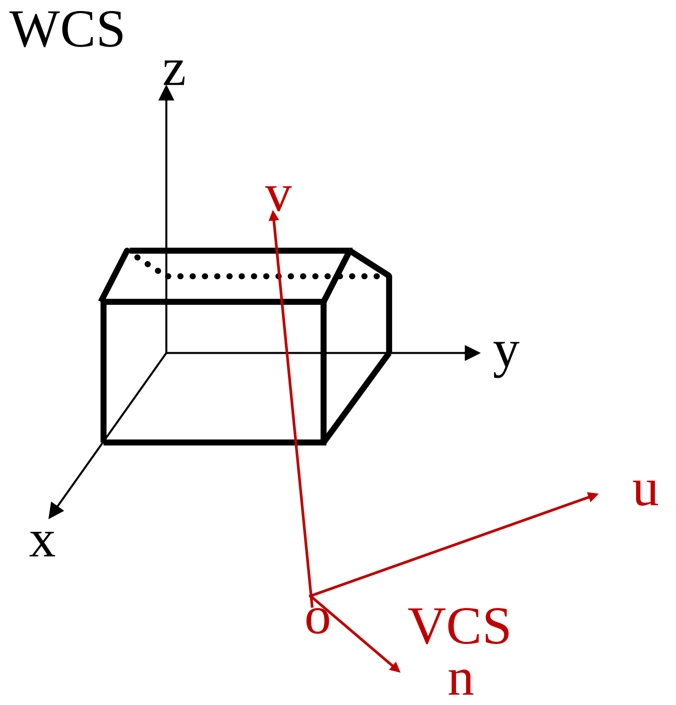
➢**3)将u，v的分量带入R矩阵**

$$R=\begin{bmatrix} 4/5, & -3/5, & 0 \\ 3/5, & 4/5, & 0 \\ 0\ , & 0\ , & 1 \end{bmatrix}$$

View Transformation（cont.)

- 3D View Transformation
  - **根据推导: $M_{wc\text{->}vc}$= R·T =R·T(-eye),**
  - **R=?   //R(u,v,n)**

$$\mathbf{R} = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



View Transformation（cont.)

➢3D View Transformation(cont.)

➢**Find R in 3D: Finding u, v, n from Eye, At and Up**

n: n和观察方向Look相反（VC是右手坐标系）

$n=N/|N|=(n_1,n_2,n_3)$ //N= - *Look* = - (At-Eye)

$$n = \frac{-Look}{\|Look\|}$$

u: 垂直于Up向量和n向量所在的平面

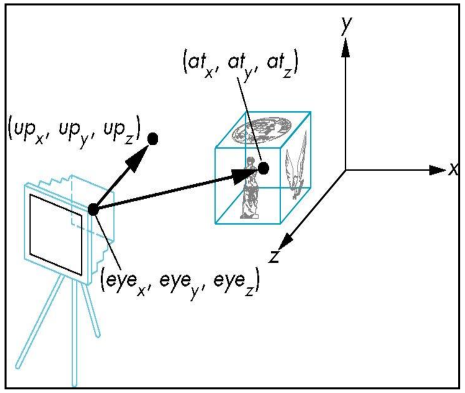
$u=(Up \times N)/|Up \times N|=(u_1,u_2,u_3)$

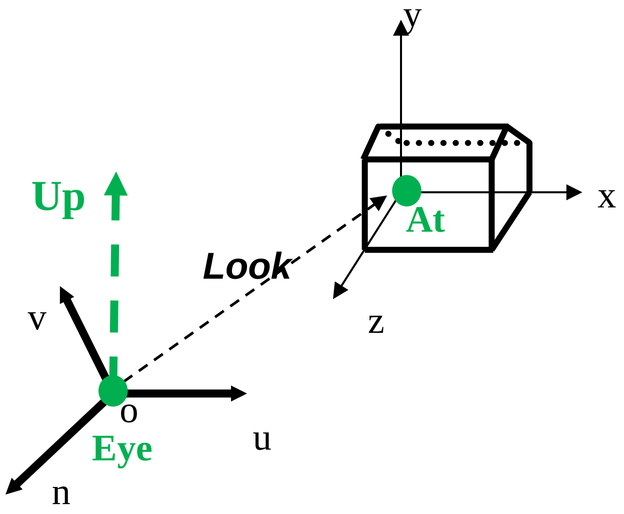
$$u = \frac{Up \times n}{\|Up \times n\|}$$

v: 通过n和u的叉乘得到

$$v = n \times u$$

$v=n \times u=(v_1,v_2,v_3)$

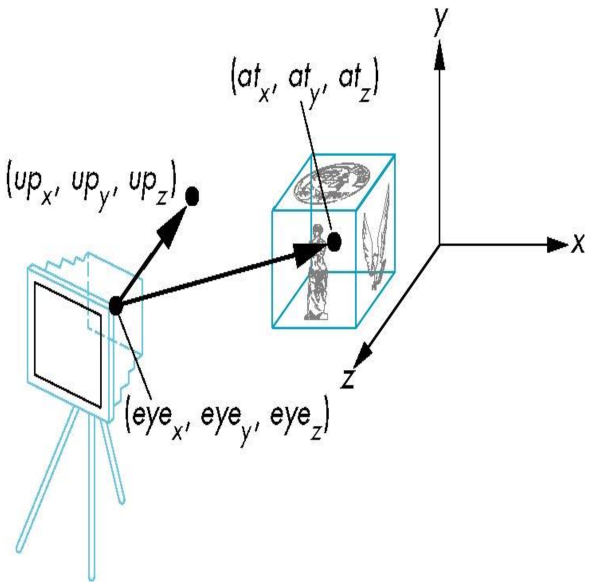




View Transformation(cont.)

- 3D View Transformation(cont.)
  - **Find R in 3D: Finding u, v, n from Eye, At and Up**

- In fact, $M_{wc\text{->}vc} = R \cdot T(-eye)$ = **lookAt(Eye, At , Up)**



```
//LookAt(eye, at, up)

var eye = vec3(1.0, 1.0, 1.0);
var at = vec3(0.0, 0.0, 0.0);
var up = vec3(0.0, 1.0, 0.0);

var mv = LookAt(eye, at, up);
```

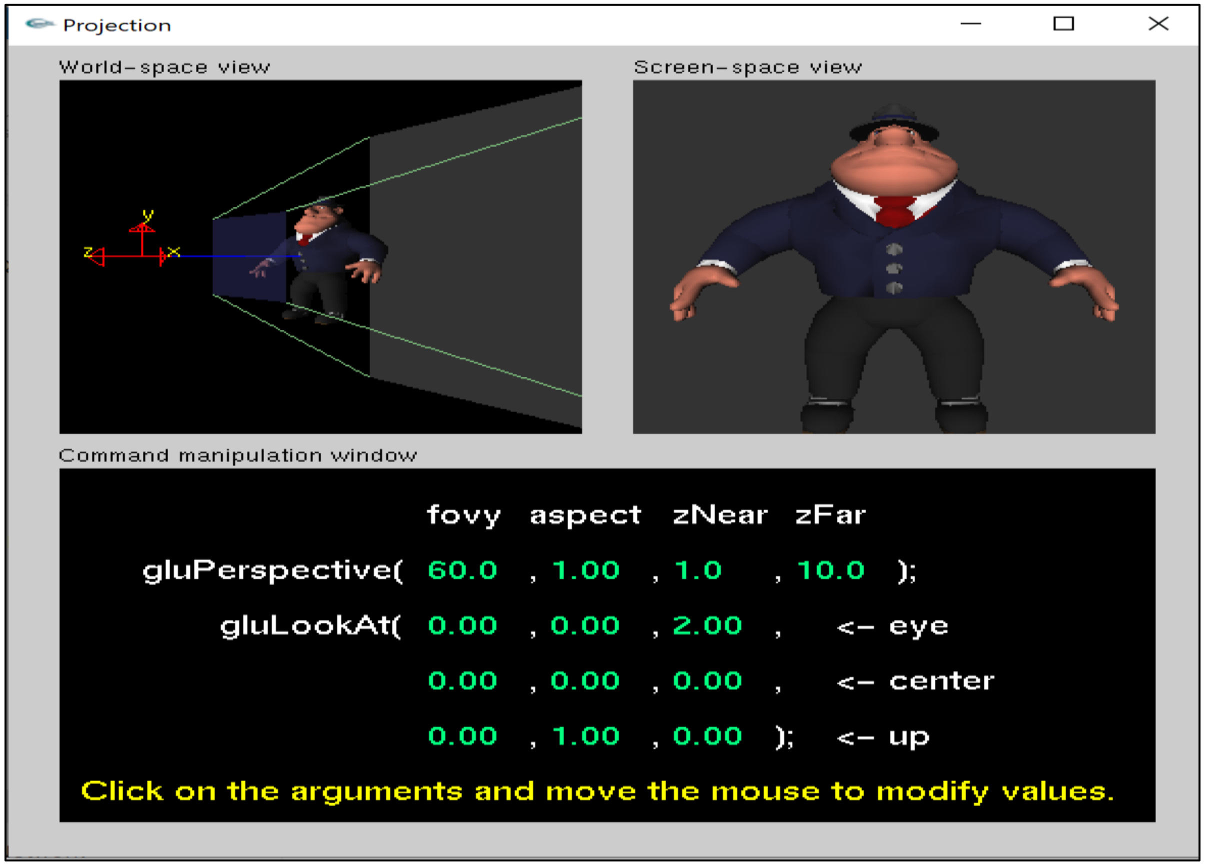
The University of New Mexico

View Transformation(cont.)

➢3D View Transformation(cont.)

➢gluLookAt (Eye,At,Up) Function (OpenGL)

- The GLU library contained the function **gluLookAt()** to form the required modelview matrix through a simple interface API function .

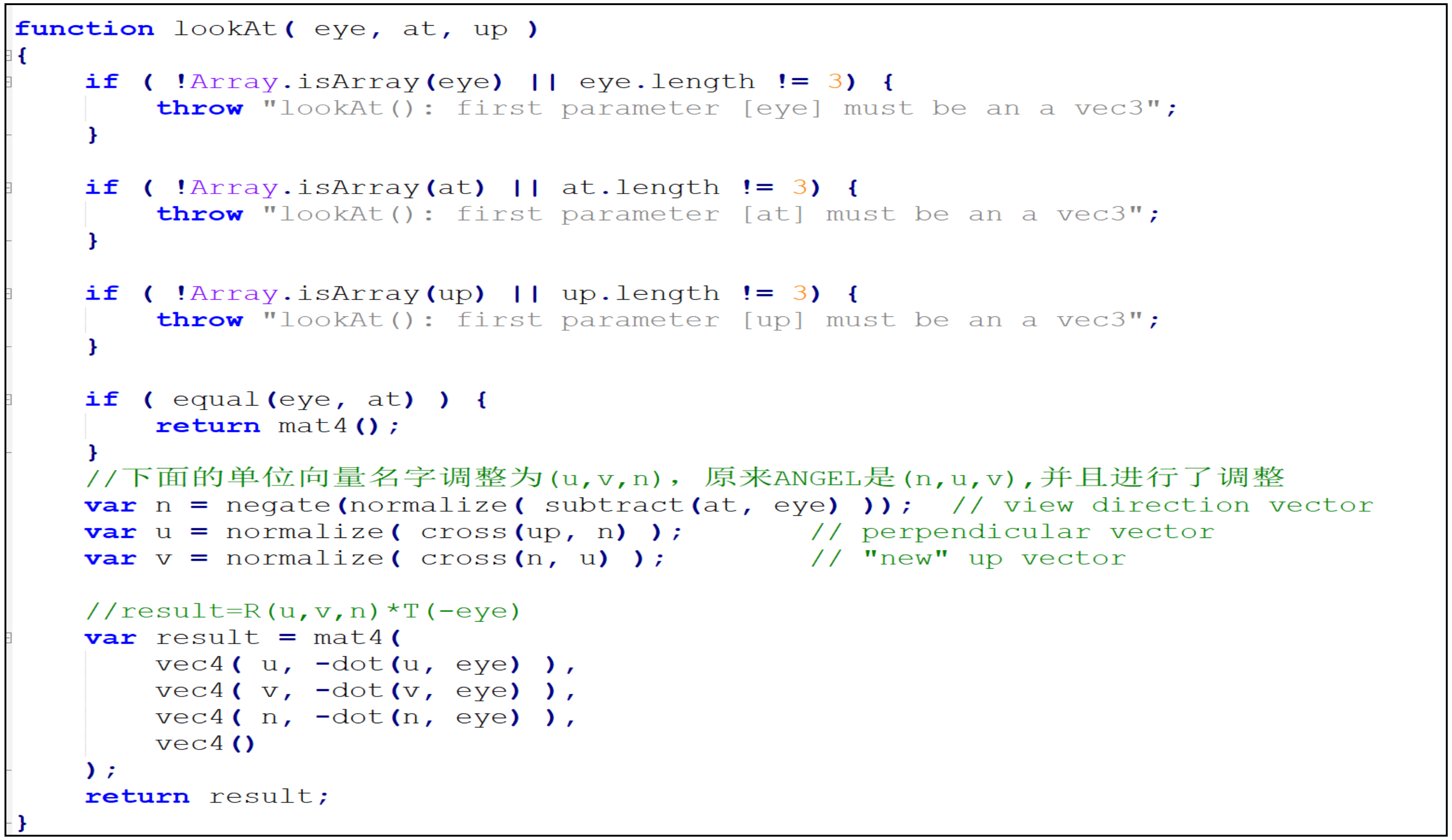


```
                 fovy   aspect   zNear   zFar
gluPerspective(  60.0  , 1.00   , 1.0   , 10.0  );
     gluLookAt(  0.00  , 0.00   , 2.00  ,      <- eye
                 0.00  , 0.00   , 0.00  ,      <- center
                 0.00  , 1.00   , 0.00  );     <- up
Click on the arguments and move the mouse to modify values.
```

View Transformation(cont.)

➢3D View Transformation(cont.)

- LookAt (Eye,At,Up) In AngelCode/common/MV.js   (webGL)



```
function lookAt( eye, at, up )
{
    if ( !Array.isArray(eye) || eye.length != 3) {
        throw "lookAt(): first parameter [eye] must be an a vec3";
    }

    if ( !Array.isArray(at) || at.length != 3) {
        throw "lookAt(): first parameter [at] must be an a vec3";
    }

    if ( !Array.isArray(up) || up.length != 3) {
        throw "lookAt(): first parameter [up] must be an a vec3";
    }

    if ( equal(eye, at) ) {
        return mat4();
    }
    //下面的单位向量名字调整为(u,v,n), 原来ANGEL是(n,u,v),并且进行了调整
    var n = negate(normalize( subtract(at, eye) ));  // view direction vector
    var u = normalize( cross(up, n) );        // perpendicular vector
    var v = normalize( cross(n, u) );         // "new" up vector

    //result=R(u,v,n)*T(-eye)
    var result = mat4(
        vec4( u, -dot(u, eye) ),
        vec4( v, -dot(v, eye) ),
        vec4( n, -dot(n, eye) ),
        vec4()
    );
    return result;
}
```

单选题 1分

设置

The University of New Mexico

4.关于 lookat(eye,at,up)描述错误的是

A 该函数用于生成“观察变换矩阵”

B 该函数内部实现等效于计算 R(u,v,n)*T(-eye)

[✓ C] 该函数实现的推导中，生成VC坐标UVN的N轴单位向量的计算公式为n=(at-eye)/|at-eye| （注：WC,VC都是右手坐标系）

D 参数up通常取向上向量（0，1，0）

提交

The University of New Mexico

Summary

- Viewing Process观察流程的级联变换

- Model-View Transformation
  - Model Transformation模型变换*
  - View Transformation观察变换*

- Projection Transformation(next class)
- Viewport Transformation(next class)

The University of New Mexico

Summary

➢观察流程的级联变换(“五”个坐标系, “四”种变换)

物体模型

▪MC模型坐标

▪将物体建模坐标描述转换到世界坐标场景下描述-模型变换-

▪WC世界坐标

▪将物体世界坐标场景描述转换为观察坐标系下描述-观察变换/相机变换

▪VC观察坐标

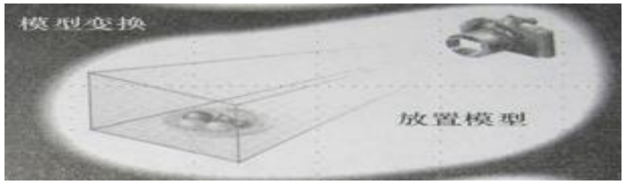
▪将观察坐标裁剪窗口内场景并进行投影变换并映射到规范化坐标下规范化区域-规范化投影变换-

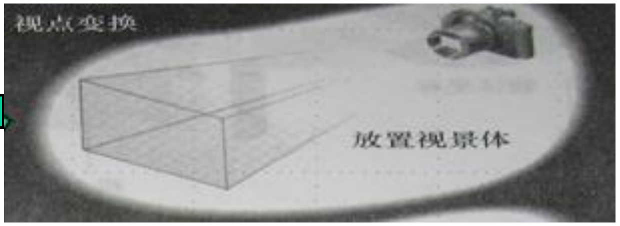
▪NDC规范化坐标

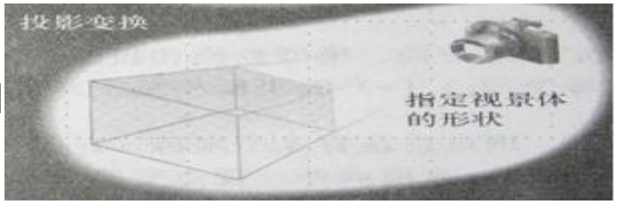
▪将规范化区域景物映射到设备坐标系下视区内-屏幕映射/视口变换

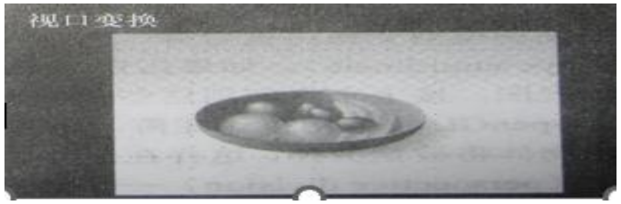
▪DC设备坐标

▪输出设备









The University of New Mexico

Summary(cont.)

- Model-View Transformation
  - Model Transformation模型变换
    - ➢Instance Transformation实例变换
      - – Defines instance transformation实例变换
        - General: M=TRS
    - ➢Complex Model Transformation复杂的模型变换
      - ➢Sun-planet-moon
      - ➢Hierarchical modeling(scene graph)

The University of New Mexico

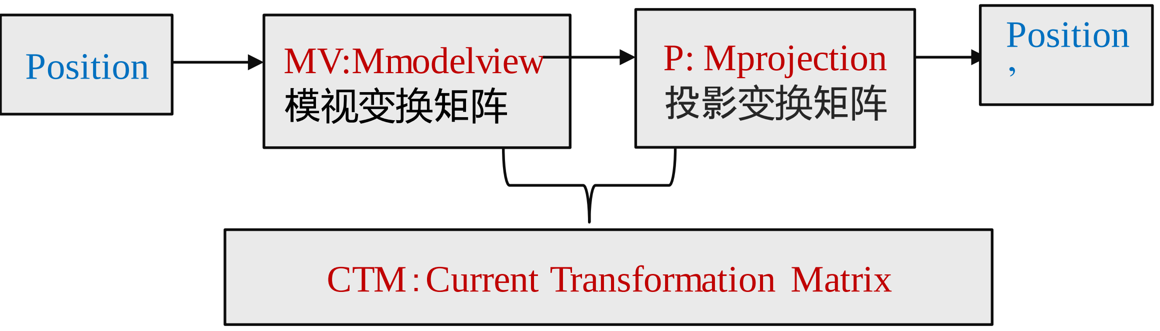
Summary(cont.)

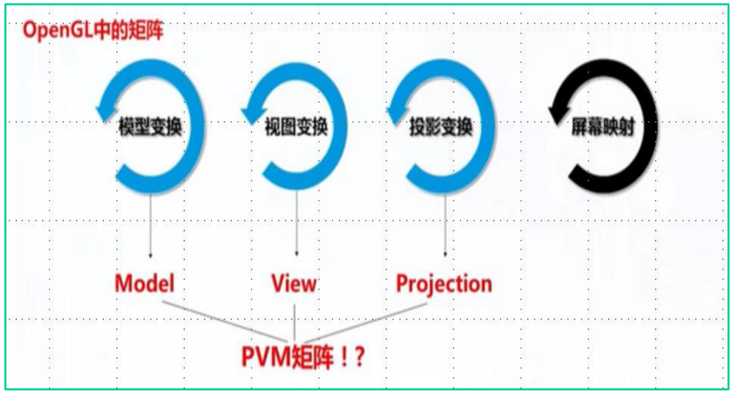
- **Model-View Transformation**
  - **观察变换(视点变换,相机变换，眼坐标变换）:在WC中建立VC，推导其 观察变换矩阵 View Matrix**
    - 思路:View Transformation is the Combination transformation that makes the VCS and WCS coincide(重合): **“move VCS to WCS”**
    - 计算方法和实现函数
      - **计算：M= R * T=R（u, v , n) * T(-eye)**
      - **函数：LookAt (eye, at, up);**

Summary(cont.)

➢观察的编程:PVM

- ➢Position’= CTM * Position; //CTM:current Transformation Matrix当前变换矩阵
- ➢CTM=P* M * V = Project * ModelView = Projection * View * Model





- 模视变换矩阵: ModelView Matrix。将几何对象的表示从建模坐标变换到照相机坐标通常是一个仿射变换矩阵
- 投影变换矩阵:Projection Matrix, 将几何对象的表示转换到裁剪坐标系。

The University of New Mexico

Summary(cont.)

- 观察的编程：PVM(续)
  - 主语言数组矩阵表示通常是"行"先序表示，但GLSL中的矩阵是"列"先序表示！
    - GLSL中的矩阵存储和OpenGL相同，都是按列存储的(column-major)。这意味着矩阵的数据是按照列的顺序连续存储在内存中的。这种存储方式决定了矩阵运算的顺序和方式。在GLSL中进行矩阵运算时，通常需要将行主序存储的矩阵转换为列主序，或者在着色器中进行适当的转置操作，以确保正确的矩阵运算结果。
    - 例如：

```
glm::mat4 ans = glm::mat4(1.0f);
ans[3][0] = xTranslationValue;
//主语言数组存储是“行先序的”，  ans[3][0]表示是第4行第1列，
//顶点着色器中矩阵是“列先序的”，  ans[3][0]表示是第4列第1行
```

The University of New Mexico

作业参考实例

- 3D View Transformation(cont.)
- 观察变化的例子：相机在球面移动，漫游观察物体

  ref: //AngelCode/Chap5/hata.html

  - 相机位置eye(x,y,z)的定位

    X=r*sin(theta)cos(phi)

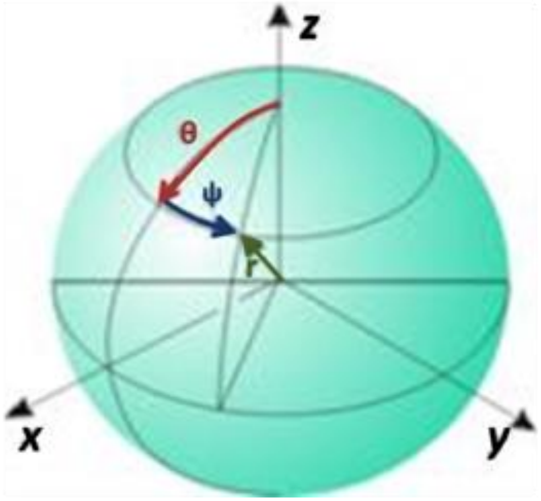
    Y=r *sin(theta)sin(phi)

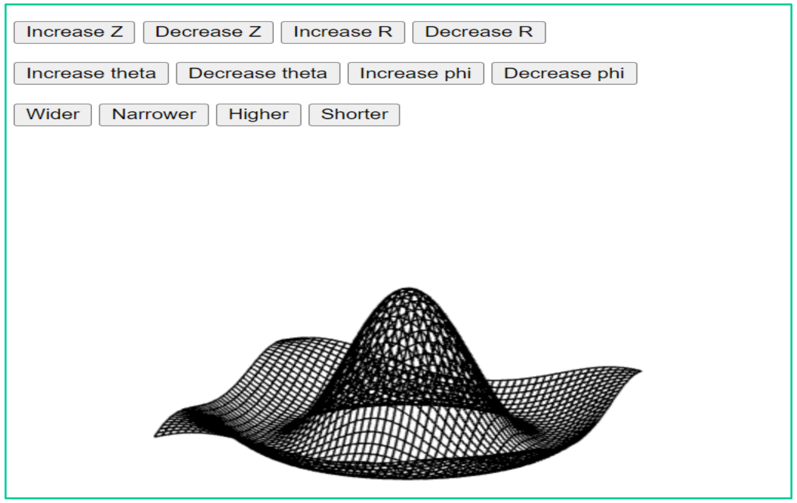
    Z=r *cos(theta)

观察变换的应用：

◆ 场景漫游：

模型变换与观察变换具有对偶性





作业参考实例(cont.)

```
window.onload = function init() {
    ……
   ViewMatrixLoc = gl.getUniformLocation(program, "ViewMatrix");
   ……}
```

```
var render = function() {
     gl.uniformMatrix4fv(ViewMatrixLoc, false, flatten(ViewMatrix) );
     ViewMatrix = lookAt(eye, at , up);
      …
      ….}
```

```
//vertex  shader
Uniform Mat4 ViewMatrix;
void main() {
      //输出的顶点坐标gl_Position，应该是裁剪坐标系下的坐标！
      gl_Position =projectionMatrix*ViewMatrix* ModelMatrix vPosition;
       ……
}
```