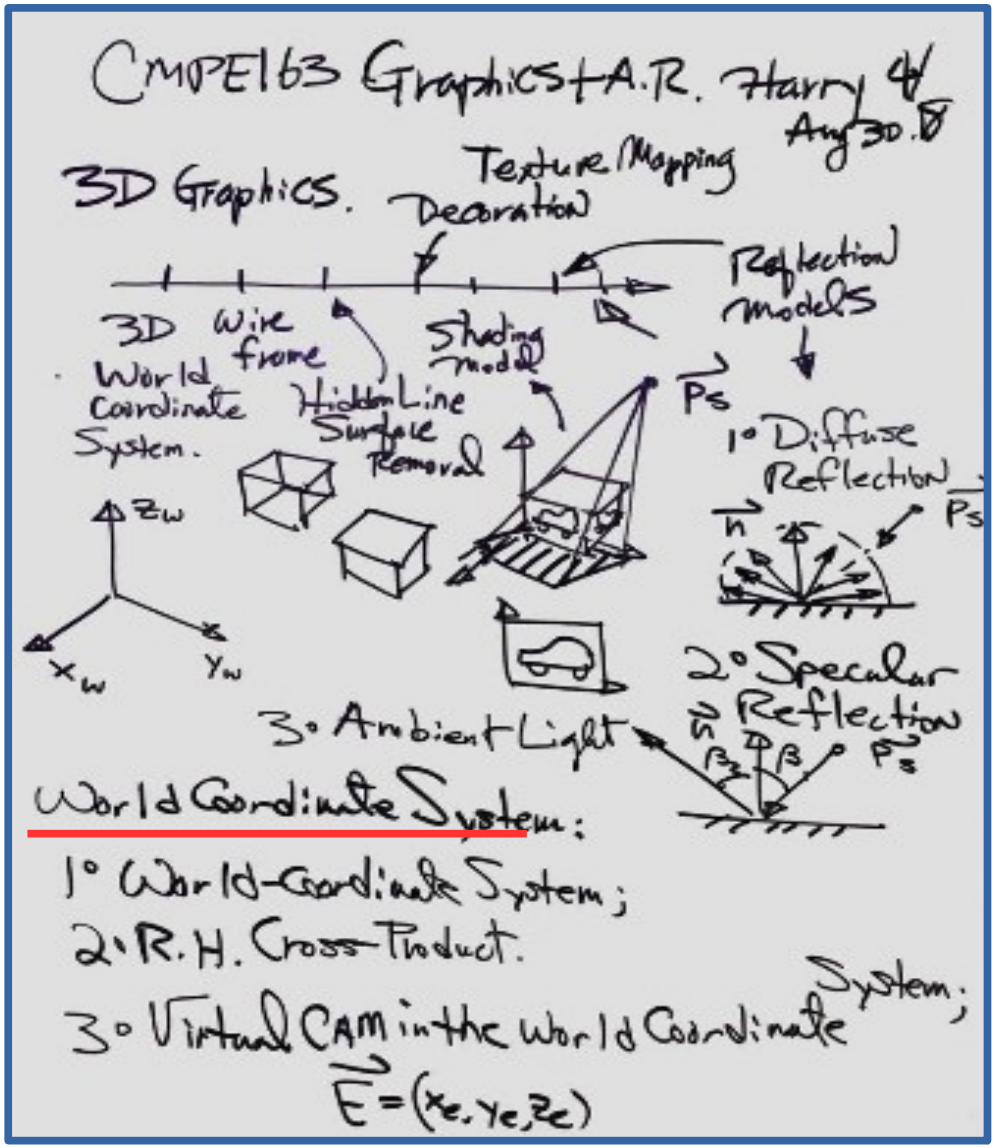
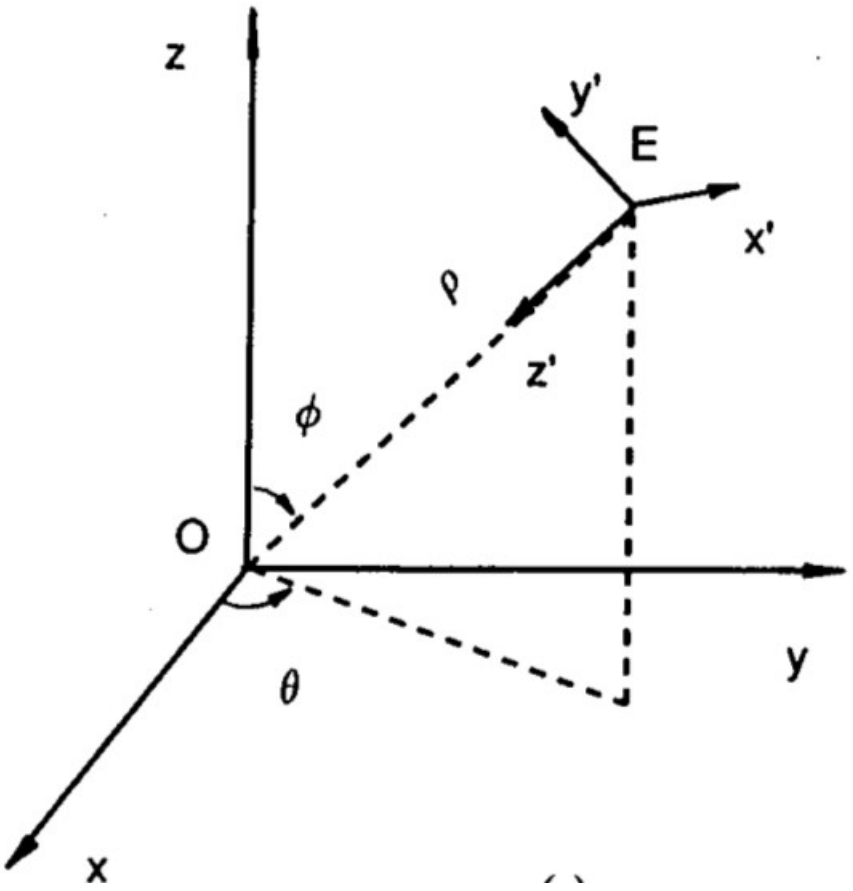
3D World Coordinate System

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992

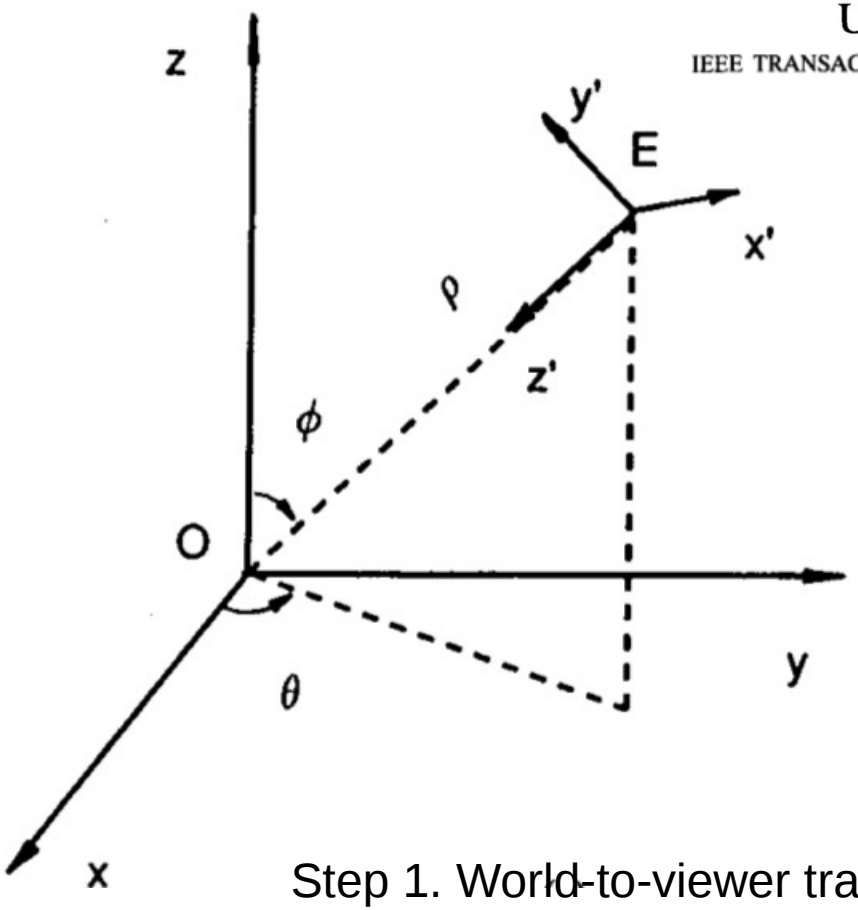




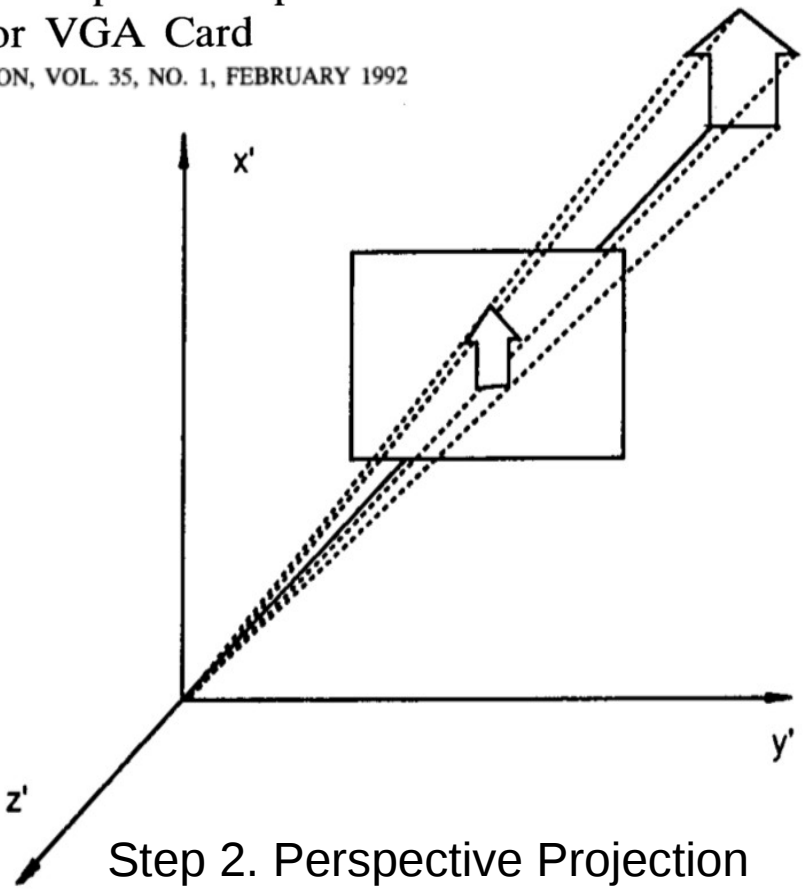
3D Transformation Pipeline Technique

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



Step 1. World-to-viewer transform



Step 2. Perspective Projection

$$\mathbf{T} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

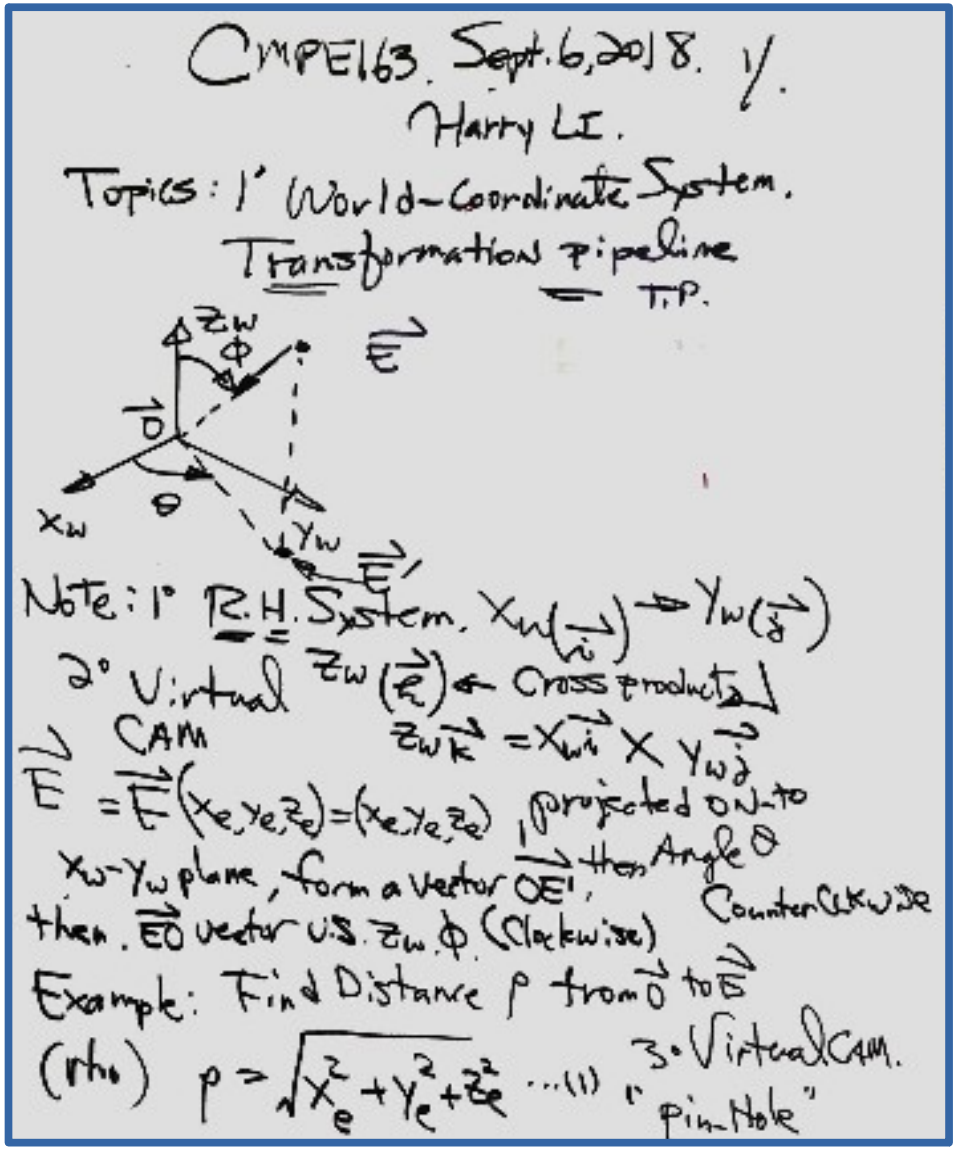
$$x_p = x_e\left(\frac{D}{z_e}\right)$$

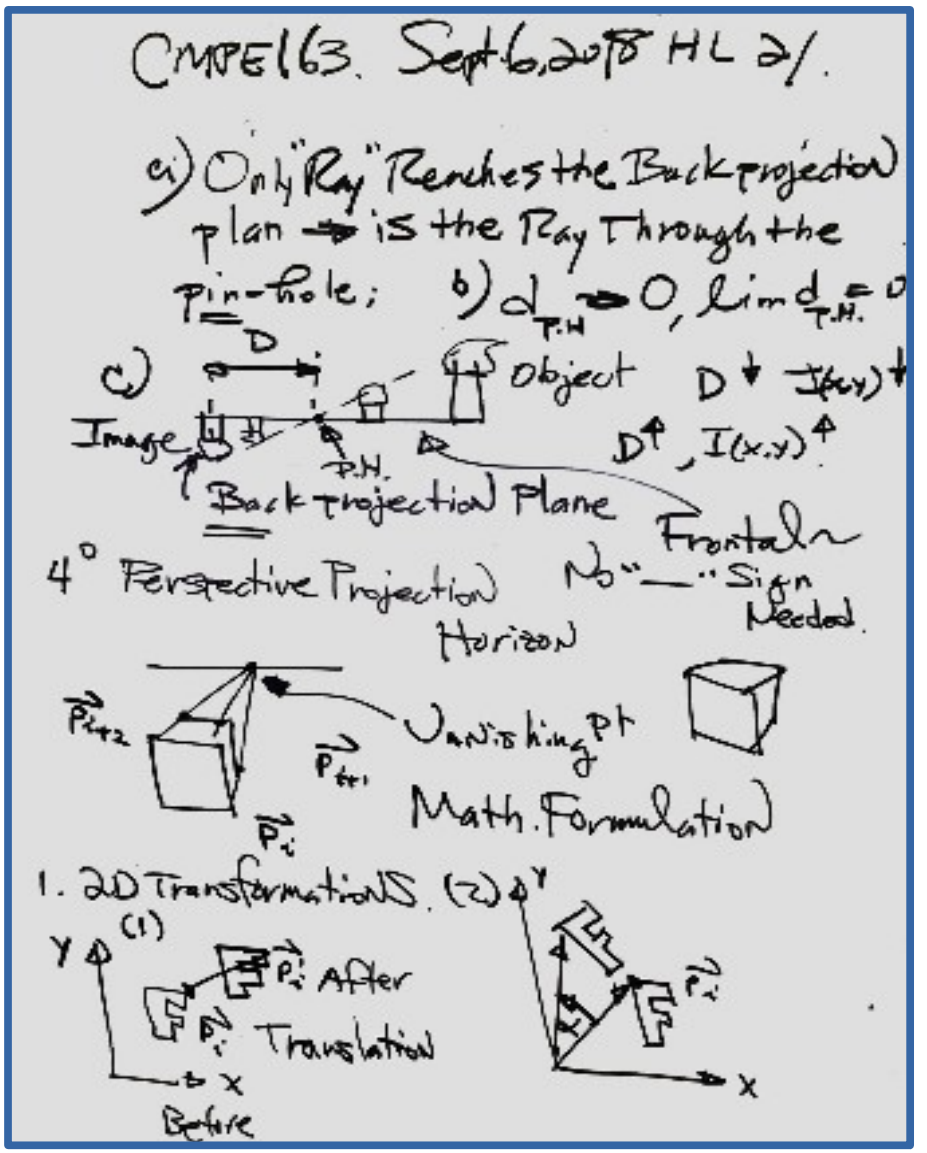
$$y_p = y_e\left(\frac{D}{z_e}\right)$$

9-6-2018: Viewer Coordinate System And Virtual Camera

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992

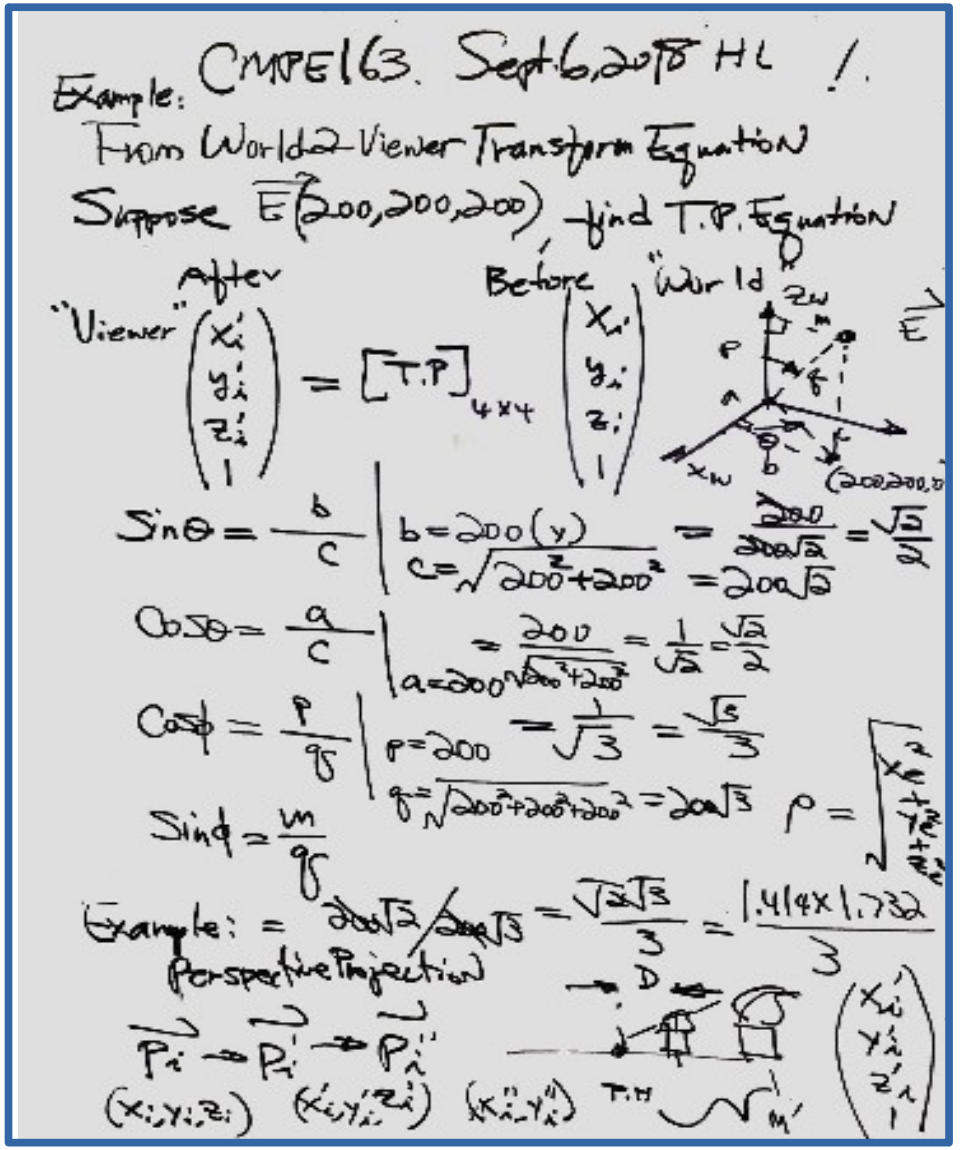




9-6-2018 World-tp-Viewer Transforms

Reference: H. Li Three-Dimensional Computer Graphics Using EGA or VGA Card

IEEE TRANSACTIONS ON EDUCATION, VOL. 35, NO. 1, FEBRUARY 1992



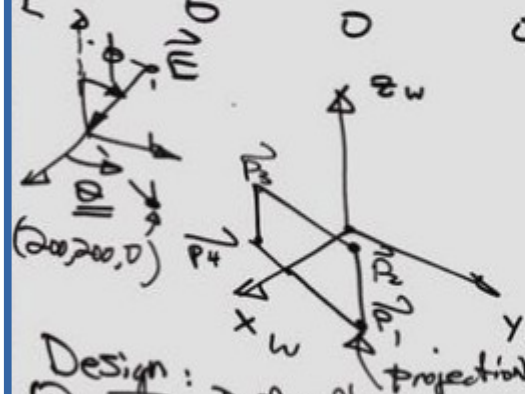
9-13-2018 Design 3D Projection In Virtual Space

CMPE163 Sept. 13, 2018. HL

Homework: Design 3D Projection system.

$$\begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \ldots(1)$$

and Compute Transformation Pipeline. plot the result



Design:

① Dataset $\{\vec{P}_i(x_i, y_i, z_i) \mid i = 1, 2, \ldots, N\}$

$\vec{P}_1(x_1, y_1, z_1) = (50, 50, \rho)$, $\vec{P}_2(x_2, y_2, z_2) = (50, 50, 100)$

$\vec{P}_3(x_3, y_3, z_3) = (50, -50, 100)$, $\vec{P}_4(x_4, y_4, z_4) = (50, -50, 0)$

② Define $\vec{E}(200, 200, 200)$, $\rho_{(\text{this})} = 200\sqrt{3}$

$\sin\theta = \frac{200}{\sqrt{200^2+200^2}} = \frac{\sqrt{2}}{2}$, $\cos\theta = \frac{\sqrt{2}}{2}$, $\sin\phi = \frac{200\sqrt{2}}{200\sqrt{3}} = \frac{\sqrt{6}}{3}$

$\cos\phi = \frac{200}{200\sqrt{3}} = \frac{\sqrt{3}}{3}$

Substitute the Calculation Result into Eqn(1), Hence, we have,

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3} & -\frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3} & \frac{\sqrt{6}}{3} & 0 \\ -\frac{\sqrt{2}}{2}\frac{\sqrt{6}}{3} & -\frac{\sqrt{2}}{2}\frac{\sqrt{6}}{3} & -\frac{\sqrt{3}}{3} & 200\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i & 50 \\ y_i & 50 \\ z_i & 100 \\ & i=2 \\ 1 & \end{bmatrix} \quad \ldots(2)$$

After (Viewer) — Before (World) "SPARSE"

Therefore, Each $\vec{P}_i$ in the World Coordinate System Can be Mapped to Viewer Coordinate System.

$\vec{P}_1'(x_1', y_1', z_1')$: From Eqn(2) for $i=1$.

$x_1' = -\frac{\sqrt{2}}{2}x_i + \frac{\sqrt{2}}{2}\cdot y_i + 0\cdot z_i + 0\cdot 1 = -\frac{\sqrt{2}}{2}\times 50 + \frac{\sqrt{2}}{2}\times 50 = 0$

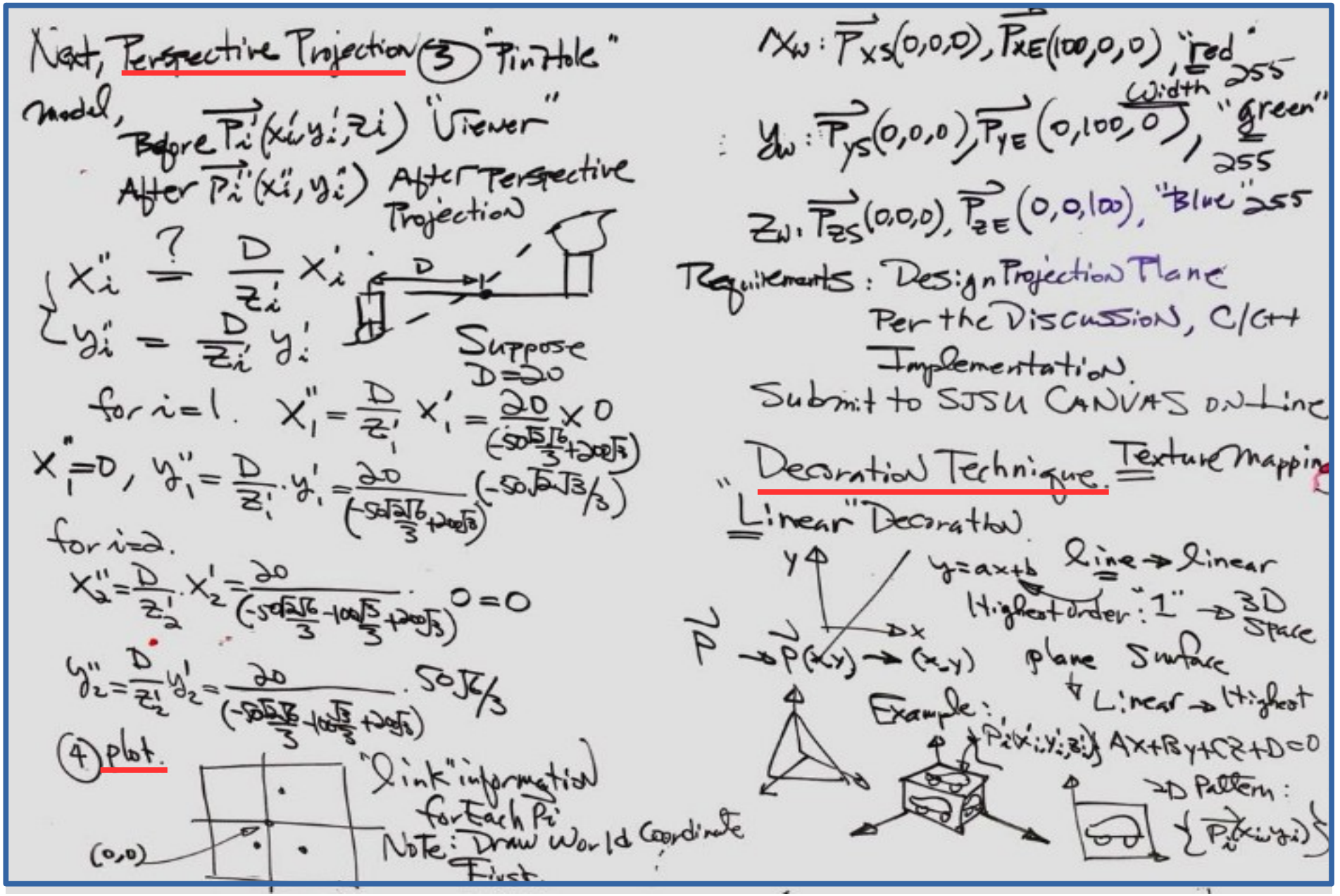
$y_1' = -\frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3}x_i - \frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3}y_i + \frac{\sqrt{6}}{3}\cdot z_i + 0\cdot 1 = -50\sqrt{2}\cdot\frac{\sqrt{3}}{3}$

$z_1' = -\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{6}}{3}\cdot x_i - \frac{\sqrt{2}}{2}\frac{\sqrt{6}}{3}y_i - \frac{\sqrt{3}}{3}z_i + 200\sqrt{3} = -50\sqrt{2}\frac{\sqrt{6}}{3} + 200\sqrt{3}$

$x_2' = -\frac{\sqrt{2}}{2}x_2 + \frac{\sqrt{2}}{2}y_2 = 0$ $\quad y_2' = -\frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3}x_i - \frac{\sqrt{2}}{2}\frac{\sqrt{3}}{3}y_i + 100\frac{\sqrt{6}}{3} = -50\frac{\sqrt{2}\sqrt{3}}{3} + 100\frac{\sqrt{6}}{3}$

$z_2' = -\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{6}}{3}x_2 - \frac{\sqrt{2}}{2}\frac{\sqrt{6}}{3}y_2 - \frac{\sqrt{3}}{3}z_2' + 200\sqrt{3} = -50\sqrt{2}\frac{\sqrt{6}}{3} - 100\frac{\sqrt{6}}{3} + 200\sqrt{3}$

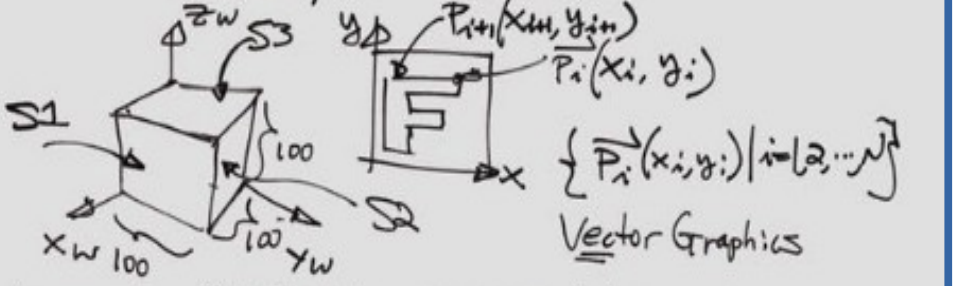
9-13-2018 Design 3D Projection In Virtual Space



9-13-2018 Linear Decoration in "World"

CMPE163 Sept. 13, 2018. HL 2/

Example: Suppose a given Cube in the "world" Coordinate System is to be decorated,



$\{\vec{P}_i(x_i, y_i) \mid i = 1, 2, \ldots N\}$

Vector Graphics

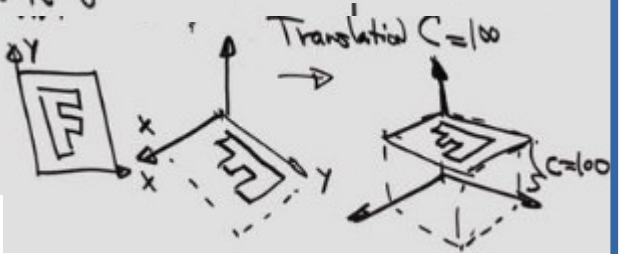
Approach: (1) Redefine 2D pattern(s) into 3D "World" Coordinate System. (2) Perform 3D Transformations (e.g. Translation, Rotation, Scaling etc) to attach to the Surface to Be decorated.

From Given 2D Pattern $\{\vec{P}_i(x_i, y_i) \mid i = 1, 2, \ldots N\}$, Re-define it in 3D.

$(X_i, Y_i) \rightarrow (x_i, y_i, C) \mid_{C=0}$

Before $P_i(x_i, y_i)$

After $P_i'(x_i', y_i', z_i')$



$$\begin{cases} x_i' = x_i \\ y_i' = y_i \\ z_i' = C \end{cases} \quad \ldots (1)$$

S1: ON y_w-z_w plane

Ind. Fn

(x_i) (y_i)

$$\begin{cases} x_i' = C \\ y_i' = x_i \\ z_i' = y_i \end{cases}$$

After. Befor.

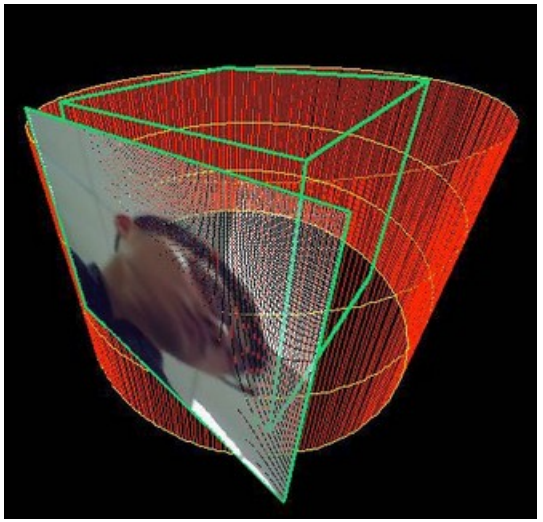
S2: ON z_w-x_w plane

Ind. Fn

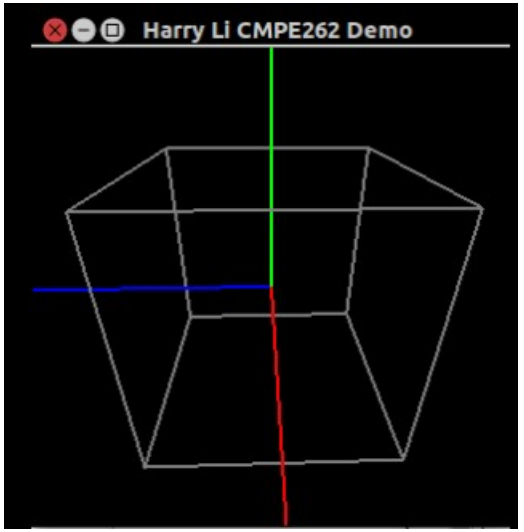
$$\begin{cases} x_i' = y_i \\ y_i' = C \\ z_i' = x_i \end{cases} \quad \ldots (2)$$

3D Transformation Pipeline Program (1)

OpenGL/lecWireframe



Create green frame above




```
/*********************************************************
 * Program: wireframe.c   for CMPE262                    *
 * Date: Sept 12, 2013                                   *
 * gcc main.cpp -o main.o -lGL -lGLU -lglut -lm    *
 * Note: linking be sure to have included math lib *
 *       e.g., -lm                                       *
 *********************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <GL/glut.h>

void Display(void);
void CreateEnvironment(void);
void MakeGeometry(void);
void MakeLighting(void);
void MakeCamera(int,int,int);
void HandleKeyboard(unsigned char key,int x, int y);
void HandleSpecialKeyboard(int key,int x, int y);
void HandleMouse(int,int,int,int);
void HandleMainMenu(int);
void HandleSpeedMenu(int);
void HandleVisibility(int vis);
void HandleIdle(void);
void DrawTextXY(double,double,double,double,char *);
void GiveUsage(char *);
```

3D Transformation Pipeline Program (2)

```
#define TRUE  1
#define FALSE 0
#define PI 3.14159265358979323846264643
#define DRAFT  0
#define MEDIUM 1
#define BEST   2

int drawquality = DRAFT;
int spincamera = TRUE;
int cameradirection = 1;
double updownrotate = 60;
int ballbounce = TRUE;
double ballspeed = 2;

#define OVALID      1
#define SPHEREID    2
#define BOXID       3
#define PLANEID     4
#define TEXTID      5
```

```
int main(int argc,char **argv)
{
   int i,j,depth;
   int mainmenu,speedmenu;

   for (i=1;i<argc;i++) {
      if (strstr(argv[i],"-h") != NULL)
         GiveUsage(argv[0]);
      if (strstr(argv[i],"-q") != NULL)
{
         if (i+1 >= argc)
            GiveUsage(argv[0]);
         drawquality = atoi(argv[i+1]);
         if (drawquality < DRAFT)
            drawquality = DRAFT;
         if (drawquality > BEST)
            drawquality = BEST;
         i++;
      }
   }
```

3D Transformation Pipeline Program (3)

```
  /* Set things up and go */
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_DOUBLE |
                      GLUT_RGB | GLUT_DEPTH);
  glutCreateWindow("Harry Li CMPE262 Demo");
  glutDisplayFunc(Display);
  glutVisibilityFunc(HandleVisibility);
  glutKeyboardFunc(HandleKeyboard);
  glutSpecialFunc(HandleSpecialKeyboard);
  glutMouseFunc(HandleMouse);

  CreateEnvironment();

/* Set up some menus */
  speedmenu = glutCreateMenu(HandleSpeedMenu);
  glutAddMenuEntry("Slow",1);
  glutAddMenuEntry("Medium",2);
  glutAddMenuEntry("fast",3);
  mainmenu = glutCreateMenu(HandleMainMenu);
  glutAddMenuEntry("Toggle camera spin",1);
  glutAddMenuEntry("Toggle ball bounce",2);
  glutAddSubMenu("Ball speed",speedmenu);
  glutAddMenuEntry("Quit",100);
  glutAttachMenu(GLUT_RIGHT_BUTTON);

  glutMainLoop();
  return(0);
}
```

```
/*************************************************
  This is where global settings are made, that is,
  things that will not change in time
*************************************************/
void CreateEnvironment(void)
{
  glEnable(GL_DEPTH_TEST);
  if (drawquality == DRAFT) {
    glShadeModel(GL_FLAT);
  }
  if (drawquality == MEDIUM) {
    glShadeModel(GL_SMOOTH);
  }

  if (drawquality == BEST) {
    glEnable(GL_LINE_SMOOTH);
    glEnable(GL_POINT_SMOOTH);
    glEnable(GL_POLYGON_SMOOTH);
    glShadeModel(GL_SMOOTH);
    glDisable(GL_DITHER);  /* Assume RGBA  */
  }
  glLineWidth(1.0);
  glPointSize(1.0);
  glPolygonMode(GL_FRONT_AND_BACK,GL_FILL);
  glFrontFace(GL_CW);
  glDisable(GL_CULL_FACE);
  glClearColor(0.0,0.0,0.0,0.0);        /* Background colour */
  glEnable(GL_COLOR_MATERIAL);
}
```

3D Transformation Pipeline Program (4)

```
   /* Place a few grey boxes around the place */
   glLoadName(BOXID);
   glColor3f(0.5,0.5,0.5);
   if (drawquality > DRAFT) {
  glMaterialfv(GL_FRONT_AND_BACK,
                    GL_DIFFUSE,mdiff3);
  glMaterialfv(GL_FRONT_AND_BACK,
                    GL_AMBIENT,mamb3);
   }
   glPushMatrix();
 //  glTranslatef(1.8,0.2,1.8);
glTranslatef(0,0,0);
   if (drawquality > DRAFT)
     glutSolidCube(200);
   else
      glutWireCube(200);
  /* glTranslatef(-3.6,0.0,0.0);
   if (drawquality > DRAFT)
     glutSolidCube(0.4);
   else
      glutWireCube(0.4);*/
   glPopMatrix();
// Harry Li, 2013-9-12
}
/***********************************************
   Set up the lighing environment
***********************************************/
```

```
 void MakeLighting(void)
{
   GLfloat globalambient[] = {0.3,0.3,0.3,1.0};

   /* The specifications for 3 light sources */
   GLfloat pos0[] = {1.0,1.0,0.0,0.0};   /* w = 0 == infinite distance */
   GLfloat dif0[] = {0.8,0.8,0.8,1.0};

   GLfloat pos1[] = {5.0,-5.0,0.0,0.0};   /* Light from below */
   GLfloat dif1[] = {0.4,0.4,0.4,1.0};      /* Fainter */

   if (drawquality > DRAFT) {

      /* Set ambient globally, default ambient for light sources is 0 */
      glLightModelfv(GL_LIGHT_MODEL_AMBIENT,globalambient);

      glLightfv(GL_LIGHT0,GL_POSITION,pos0);
      glLightfv(GL_LIGHT0,GL_DIFFUSE,dif0);

      glLightfv(GL_LIGHT1,GL_POSITION,pos1);
      glLightfv(GL_LIGHT1,GL_DIFFUSE,dif1);

      glEnable(GL_LIGHT0);
      glEnable(GL_LIGHT1);
      glEnable(GL_LIGHTING);
   }
}
```

3D Transformation Pipeline Program (5)

```
/*************************************************************
   This is the basic display callback routine
   It creates the geometry, lighting, and viewing position
   In this case it rotates the camera around the scene
*************************************************************/
void Display(void)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glPushMatrix();
   MakeCamera(FALSE,0,0);
   MakeLighting();
   MakeGeometry();
   glPopMatrix();
   /* glFlush(); This isn't necessary for double buffers */
   glutSwapBuffers();
}

/*************************************************************
   Create the geometry
*************************************************************/
void MakeGeometry(void)
{
   int i;
   double radius = 0.5;
   static double theta = 0;
   GLfloat mshin1[] = {5.0};              /* For the sphere */
   GLfloat mspec1[] = {0.5,0.5,0.5,1.0};
   GLfloat mdiff1[] = {0.6,0.0,0.6,1.0};
   GLfloat mamb1[]  = {0.1,0.0,0.1,1.0};
   GLfloat mdiff2[] = {0.0,1.0,0.0,1.0};
                           /* Green plane */
   GLfloat mamb2[]  = {0.0,0.2,0.0,1.0};
   GLfloat mdiff3[] = {0.5,0.5,0.5,1.0};
                            /* Grey boxes */
   GLfloat mamb3[]  = {0.2,0.2,0.2,1.0};
   float ORG[3] = {0,0,0};
   float XP[3] = {500,0,0}, XN[3] = {-1,0,0};
   float YP[3] = {0,500,0}, YN[3] = {0,-1,0};
   float ZP[3] = {0,0,500}, ZN[3] = {0,0,-1};

/* Create a RGB xyz axis */
// glClear(GL_CLEAR_COLOR_BUFFER_BIT
                | GL _DEPTH_BUFFER_BIT);
glLineWidth (2.0);

glBegin (GL_LINES);
glColor3f (1,0,0); // X axis is red.
glVertex3fv (ORG);
glVertex3fv (XP );
glColor3f (0,1,0); // Y axis is green.
glVertex3fv (ORG);
glVertex3fv (YP );
glColor3f (0,0,1); // z axis is blue.
glVertex3fv (ORG);
glVertex3fv (ZP );
glEnd();
```

3D Transformation Pipeline Program (6)

```
/*****************************************************
   Set up the camera
   Optionally creating a small viewport about
   the mouse click point for object selection
*****************************************************/
void MakeCamera(int pickmode,int x,int y)

{
   static double theta = 0;
   GLint viewport[4];

   /* Camera setup */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (pickmode == TRUE) {
      glGetIntegerv(GL_VIEWPORT,viewport);
                        /* Get the viewport bounds */
      gluPickMatrix(x,viewport[3]-y,3.0,3.0,viewport);
   }
   gluPerspective(70.0,    /* Field of view */
                  1.0,              /* aspect ratio  */
                  0.1,1000.0);  /* near and far  */

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(300*cos(theta*PI/180)*
                  sin(updownrotate*PI/180),
                  300*cos(updownrotate*PI/180),
                  300*sin(theta*PI/180)*
                  sin(updownrotate*PI/180),
                  0.0,0.0,0.0,                  /* Focus    */
              0.0,1.0,0.0);                     /* Up       */
     if (spincamera)
        theta += (cameradirection * 0.2);
}
/*****************************************************
   Deal with plain key strokes
*****************************************************/
void HandleKeyboard(unsigned char key,int x, int y)
{
   switch (key) {
   case 27: /* ESC */
   case 'Q':
   case 'q': exit(0); break;
   case 's':
   case 'S': spincamera = !spincamera; break;
   case 'b':
   case 'B': ballbounce = !ballbounce; break;
   }
}
```

3D Transformation Pipeline Program (7)

```
/*******************************************************
   Deal with special key strokes
*******************************************************/
void HandleSpecialKeyboard(int key,int x, int y)
{
   switch (key) {
   case GLUT_KEY_LEFT:
           cameradirection = -1; break;
   case GLUT_KEY_RIGHT:
           cameradirection = 1;  break;
   case GLUT_KEY_UP:
           updownrotate -= 2;  break;
   case GLUT_KEY_DOWN:
           updownrotate += 2;  break;
   }
}
/*******************************************************
   Handle mouse events
*******************************************************/
void HandleMouse(int button,int state,int x,int y)
{
   int i,maxselect = 100,nhits = 0;
   GLuint selectlist[100];
   if (state == GLUT_DOWN) {
      glSelectBuffer(maxselect,selectlist);
      glRenderMode(GL_SELECT);
      glInitNames();
      glPushName(-1);
```

```
glPushMatrix();
      MakeCamera(TRUE,x,y);
      MakeGeometry();
      glPopMatrix();
      nhits = glRenderMode(GL_RENDER);

      if (button == GLUT_LEFT_BUTTON) {

      } else if (button == GLUT_MIDDLE_BUTTON) {

      } /* Right button events are passed to menu handlers */

      if (nhits == -1)
         fprintf(stderr,"Select buffer overflow\n");

      if (nhits > 0) {
         fprintf(stderr,"\tPicked %d objects: ",nhits);
         for (i=0;i<nhits;i++)
            fprintf(stderr,"%d ",selectlist[4*i+3]);
         fprintf(stderr,"\n"); }

   }
}
```

3D Transformation Pipeline Program (8)

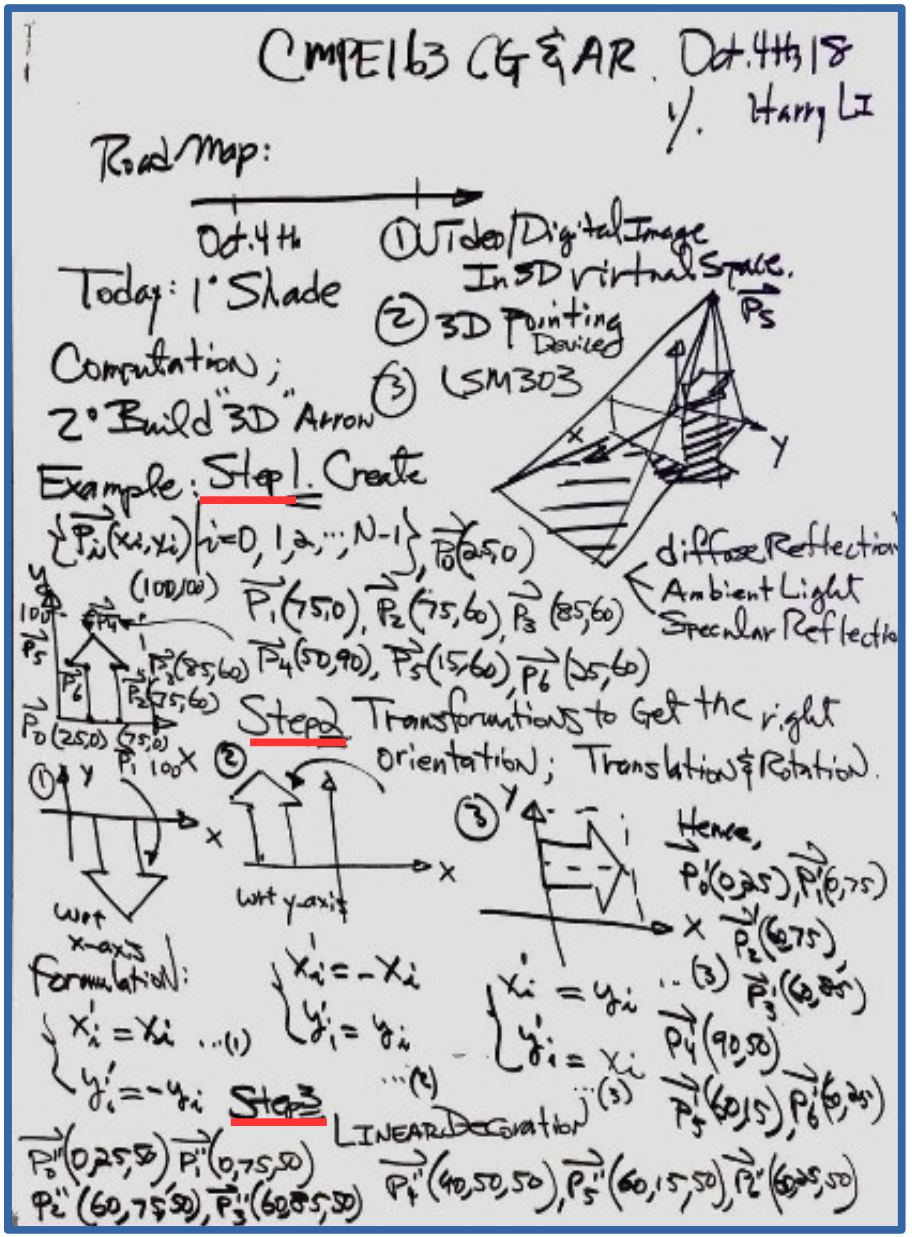
```
/*************************************************
   Handle the main menu
*************************************************/
void HandleMainMenu(int whichone)
{
   switch (whichone) {
   case 1: spincamera = !spincamera; break;
   //case 2: ballbounce = !ballbounce; break;
   case 100: exit(0); break;
   }
}
/*************************************************
   Handle the ball speed sub menu
*************************************************/
/*
void HandleSpeedMenu(int whichone)
{
   switch (whichone) {
   case 1: ballspeed = 0.5; break;
   case 2: ballspeed = 2;   break;
   case 3: ballspeed = 10;  break;
   }
}
*/
```

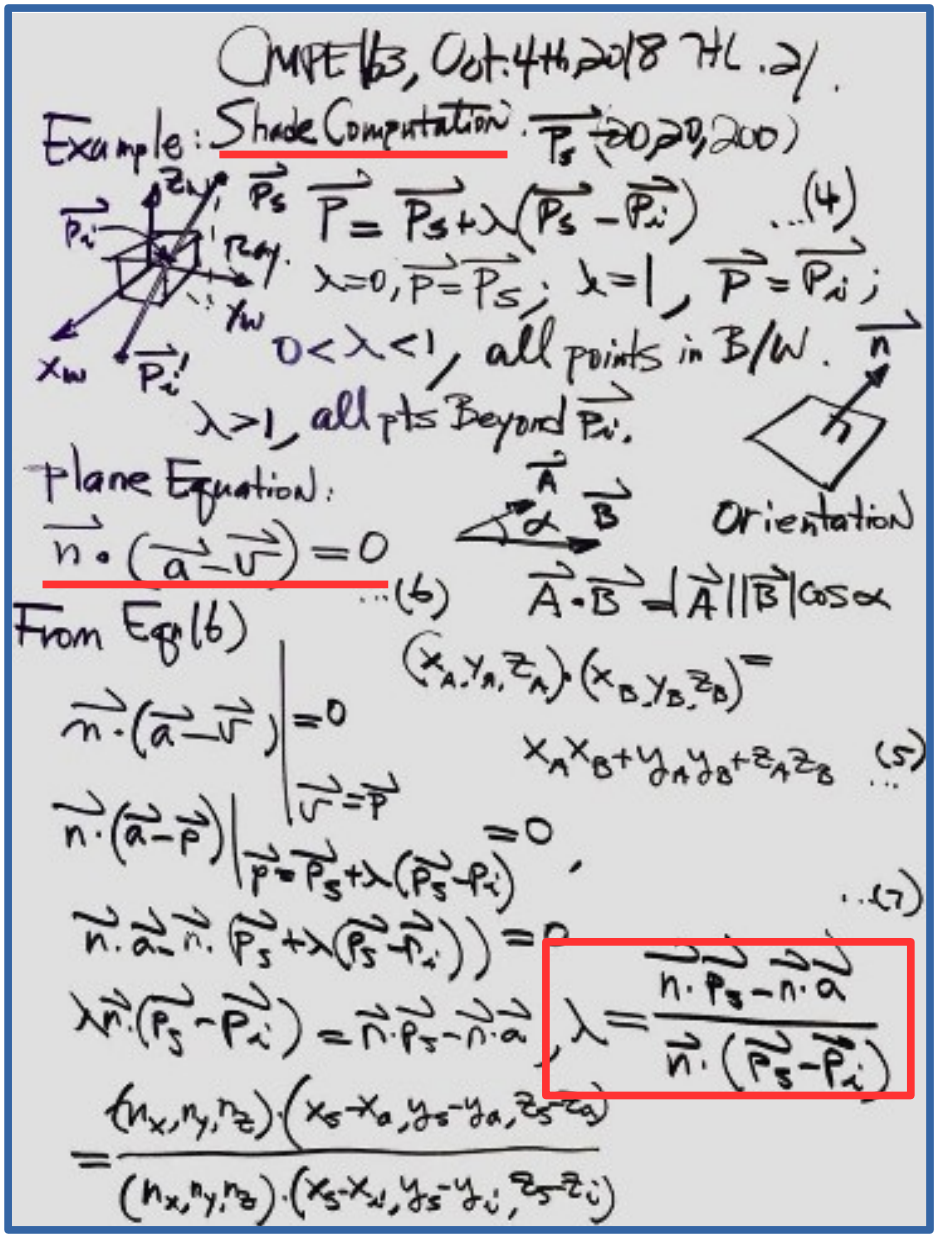
```
/*************************************************
   Handle visibility
*************************************************/
void HandleVisibility(int visible)
{
   if (visible == GLUT_VISIBLE)
      glutIdleFunc(HandleIdle);
   else
      glutIdleFunc(NULL);
}
/*************************************************
   On an idle event
*************************************************/
void HandleIdle(void)
{  glutPostRedisplay(); }
/*************************************************
   Draw text in the x-y plane
   The x,y,z coordinate is the bottom left corner
   (looking down -ve z axis)
*************************************************/
void DrawTextXY(double x,double y,double z,double scale,char *s)
{
   int i;
   glPushMatrix();
   glTranslatef(x,y,z);
   glScalef(scale,scale,scale);
   for (i=0;i<strlen(s);i++)
      glutStrokeCharacter(GLUT_STROKE_ROMAN,s[i]);
   glPopMatrix();
```

3D Transformation Pipeline Program (9)

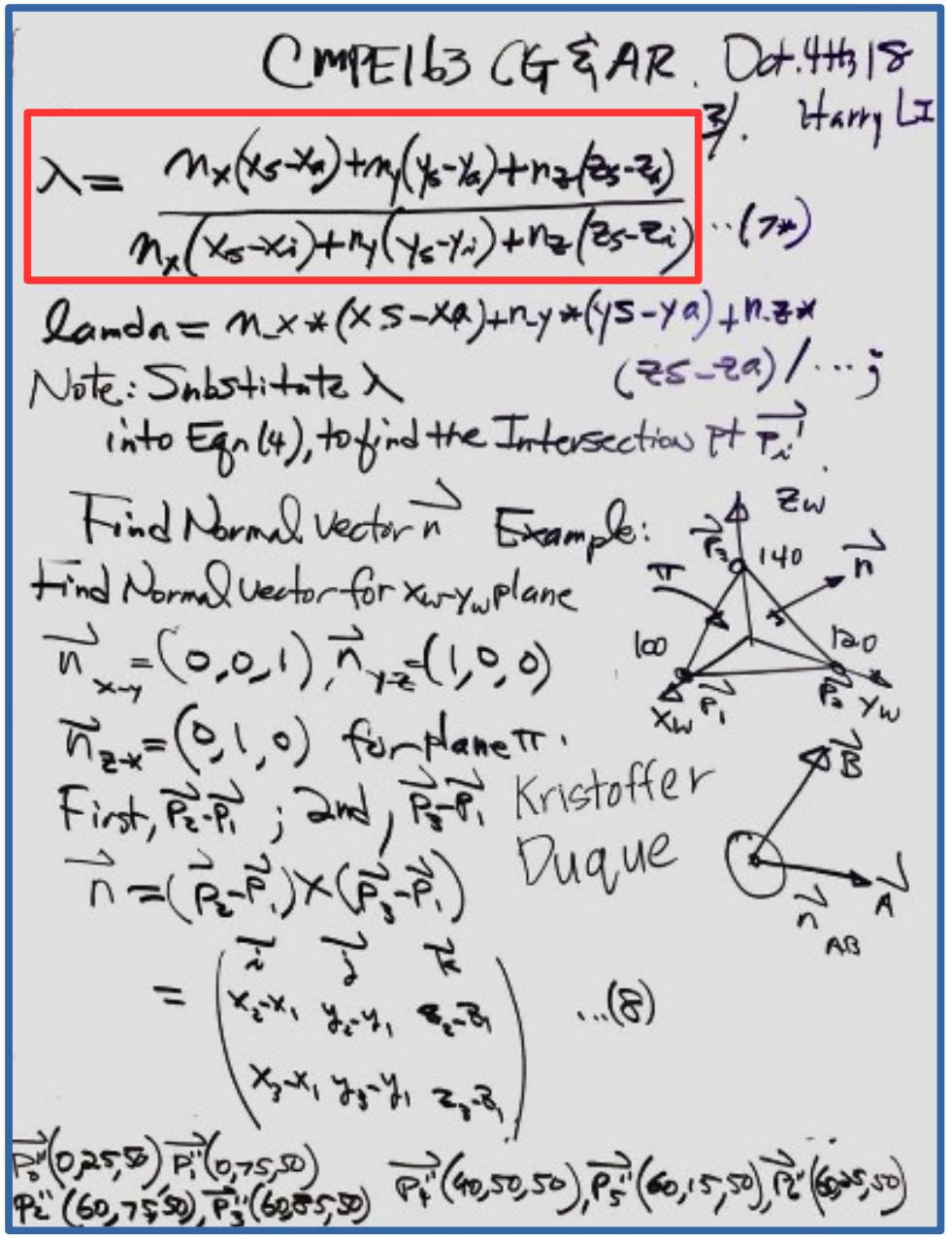
```
/*********************************************
   Display the program usage information
*********************************************/
void GiveUsage(char *cmd)
{
   fprintf(stderr,"Usage:    %s [-h] [-q n]\n",cmd);
   fprintf(stderr,"          -h   this text\n");
   fprintf(stderr,"          -q n quality, 0,1,2\n");
   fprintf(stderr,"Key Strokes and Menus:\n");
   fprintf(stderr,"           q - quit\n");
   fprintf(stderr,"           s - toggle camera spin\n");
   fprintf(stderr,"           b - toggle ball bounce\n");
   fprintf(stderr,"  left arrow - change
                   rotation direction\n");
   fprintf(stderr," right arrow - change
                   rotation direction\n");
   fprintf(stderr,"  down arrow - rotate
                   camera down\n");
   fprintf(stderr,"    up arrow - rotate
                   camera up\n");
   exit(-1);
}
```

10-4-2018 3D Projection Plane





10-4-2018 3D Projection Plane



$$\mathbf{A} = a_1\mathbf{i} + a_2\mathbf{j} + a_3\mathbf{k}$$

$$\mathbf{B} = b_1\mathbf{i} + b_2\mathbf{j} + b_3\mathbf{k}$$

$$\mathbf{A}\times\mathbf{B} = \det\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$= \mathbf{i}(a_2b_3 - a_3b_2) + \mathbf{j}(a_3b_1 - a_1b_3) + \mathbf{k}(a_1b_2 - a_2b_1)$$

Example:

$$\mathbf{A}=\mathbf{i}-\mathbf{j}$$

$$\mathbf{B}=\mathbf{i}+\mathbf{k}$$

$$\mathbf{A}\times\mathbf{B} = \mathbf{i}(-1-0) + \mathbf{j}(0-1) + \mathbf{k}(0-(-1)) = -\mathbf{i}-\mathbf{j}+\mathbf{k}$$