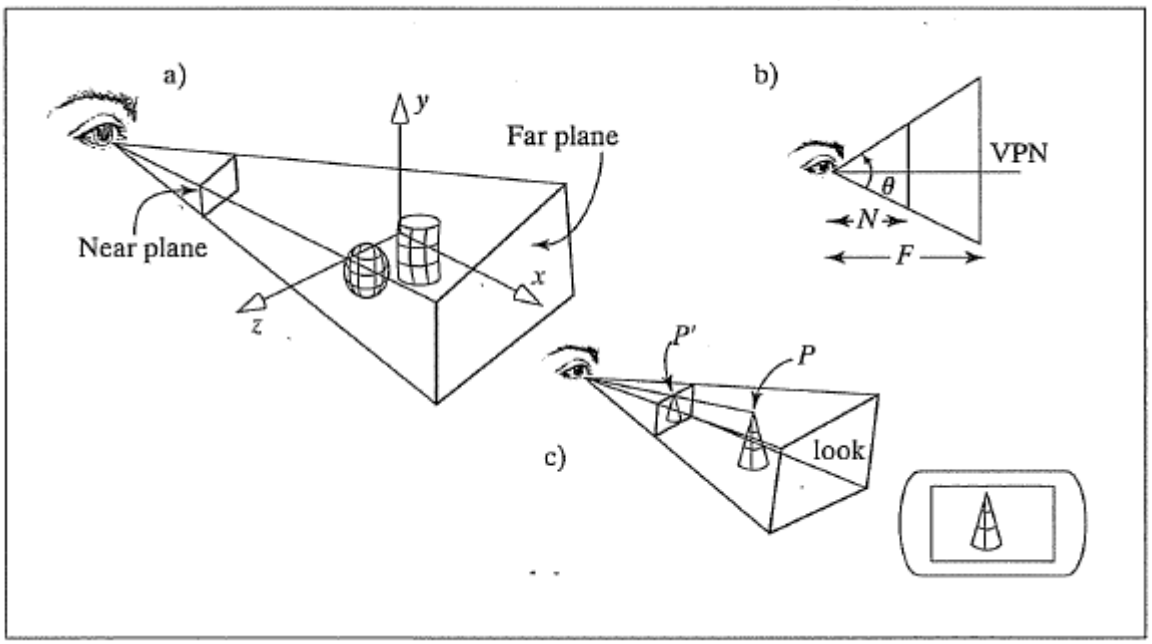
بسمه تعالی

کار با دوربین

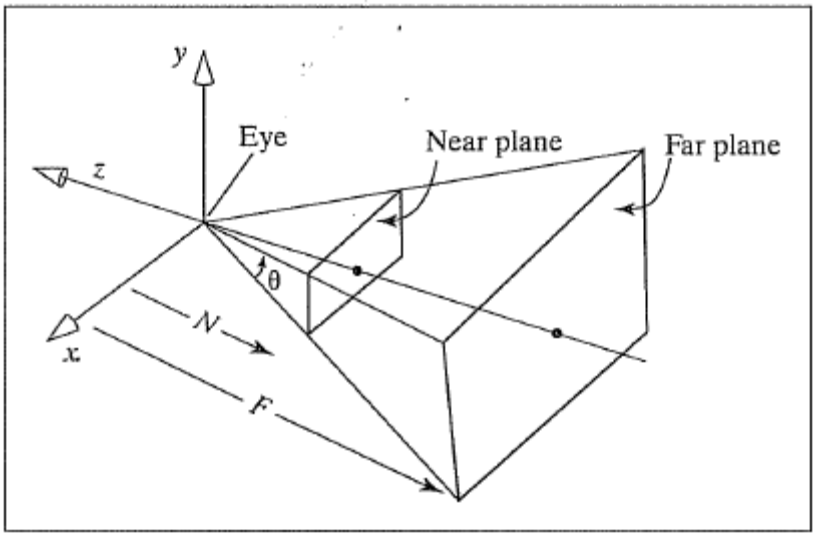
شکل ۱ یک حجم دید پرسپکتیو را نشان می دهد. صفحهٔ تصویر می تواند هر جائی بین صفحهٔ برش جلو و صفحهٔ برش عقب قرار گیرد.. شکل 1c تصویرنقطهٔ P بر روی صفحهٔ تصویر را نشان می دهد. آنچه روی صفحهٔ تصویر قرار می گیرد به بندردید برای نمایش منتقل می شود.



شکل ۱

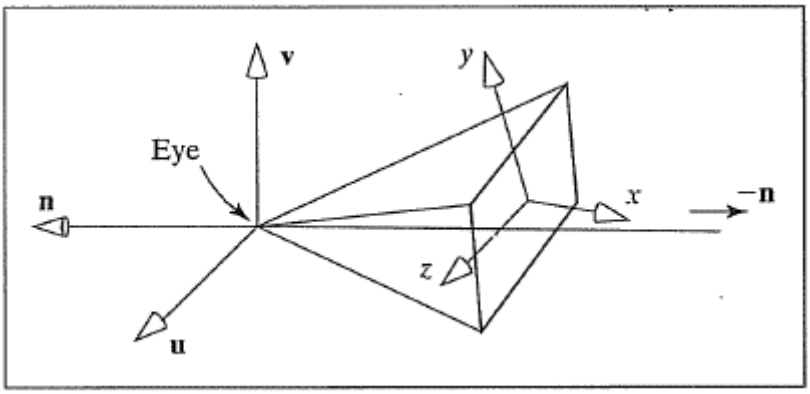
می توانیم یک دستگاه مختصات بر روی چشم (یا دوربین) در نظر بگیریم. بردار از سمت look به eye عمود به صفحهٔ تصویر VPN را می سازد.

VPN as *eye – look.*

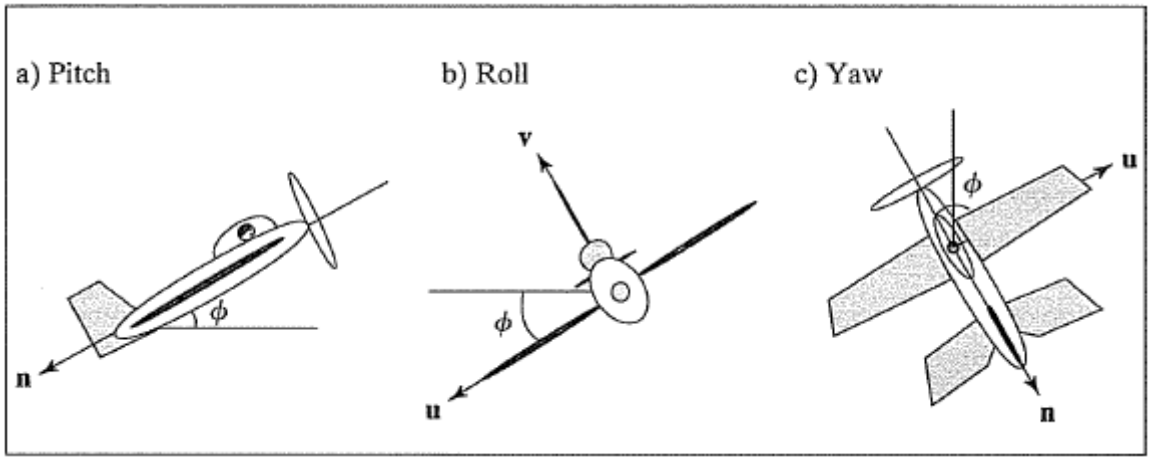


شکل ۲

eye-look را عادی می کنیم و بردار n را می سازیم. دوربین همواره به سمت n– نگاه می کند.

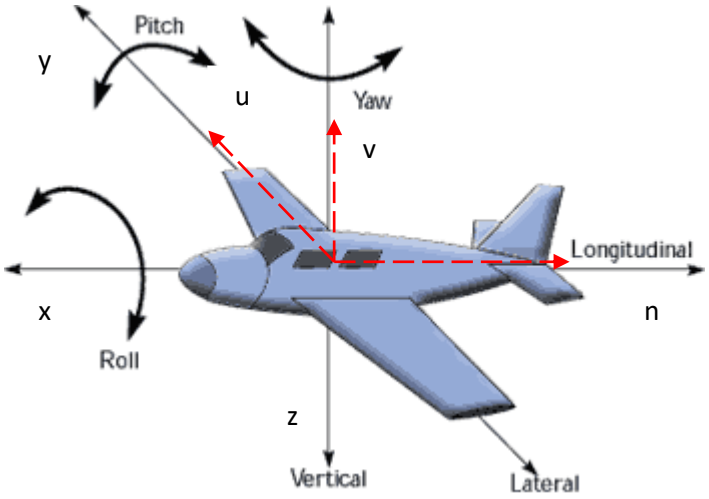


شکل ۳



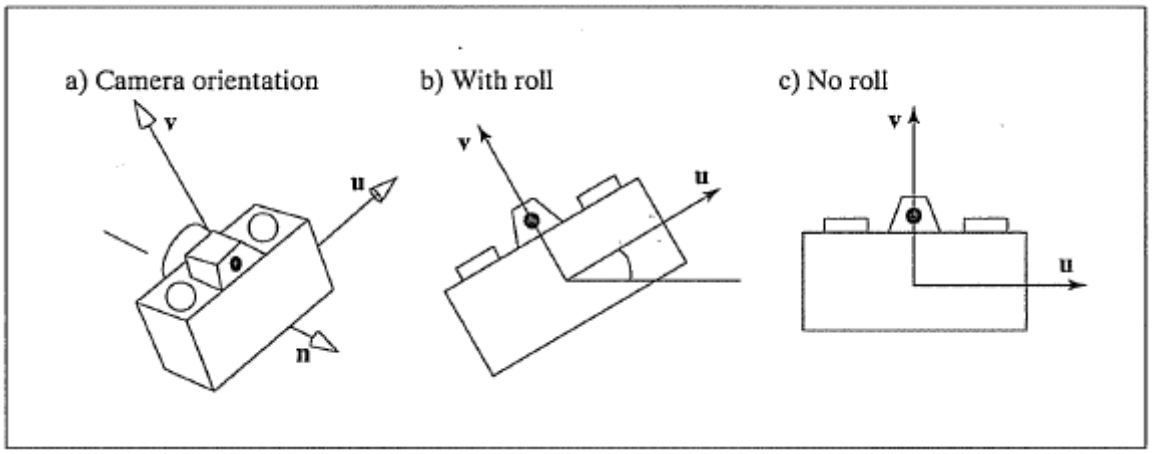
شکل ۴ توصیف تمایلهاي یک هواپیما.

در وسایل نقلیهٔ هوائی و زیردریائی محور z دستگاه مختصات به سمت مرکز کرهٔ زمین در نظر گرفته می شود. محور x برای همهٔ وسایل نقلیه در جهت حرکت آنها و در امتداد وسیله می باشد. محور y نیز با داشتن این دو محور بدست خواهد آمد. Roll دوران حول محور x، pitch دوران حول محور y، و yaw دوران حول محور z می باشد.

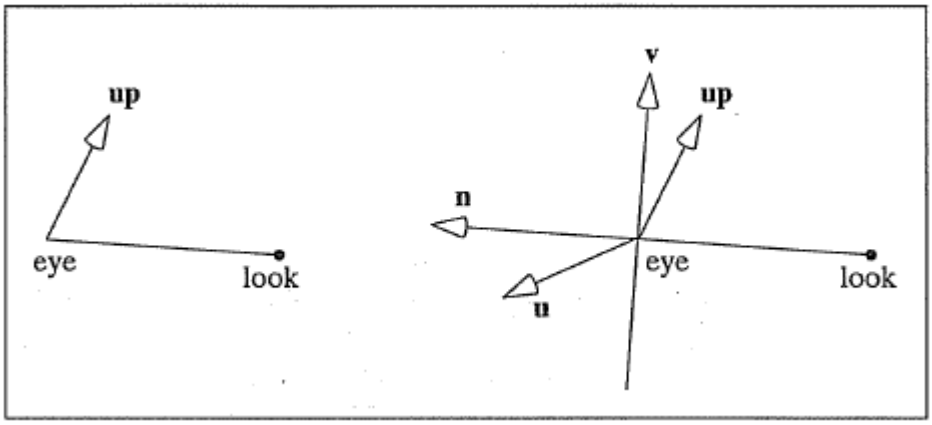


شکل ۵ تمایلهاي یک هواپیما.

حالتي مشابه با یک هواپیما را براي دوربین در نظر مي گیریم.



شکل ۶

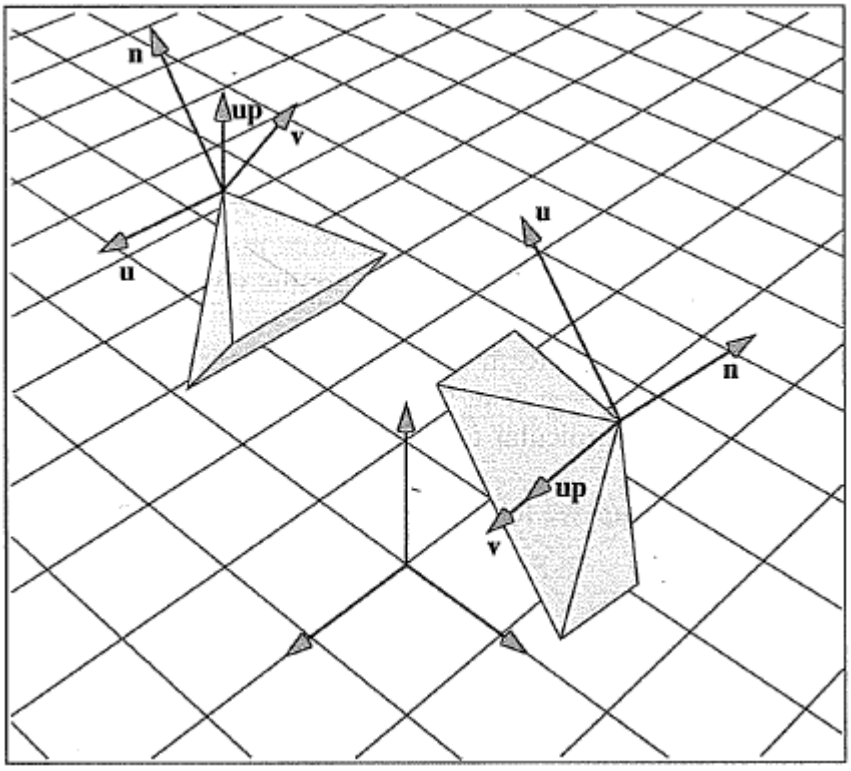


شکل ۷ ساخت بردارهاي u، v، و n.

همواره کار با glulookAt() ساده نیست و توسط آن براحتی نمی توان دوربین را در مکان و جهت دلخواه قرار داد. محورهای مختصات دوربین به روش زیر قابل دستیابی هستند.

$$\mathbf{n} = eye - look$$
$$\mathbf{u} = \mathbf{up} \times \mathbf{n}$$
$$\mathbf{v} = \mathbf{n} \times \mathbf{u}$$

البته همهٔ این بردارها باید یکه شوند. مثال:



شکل ۸

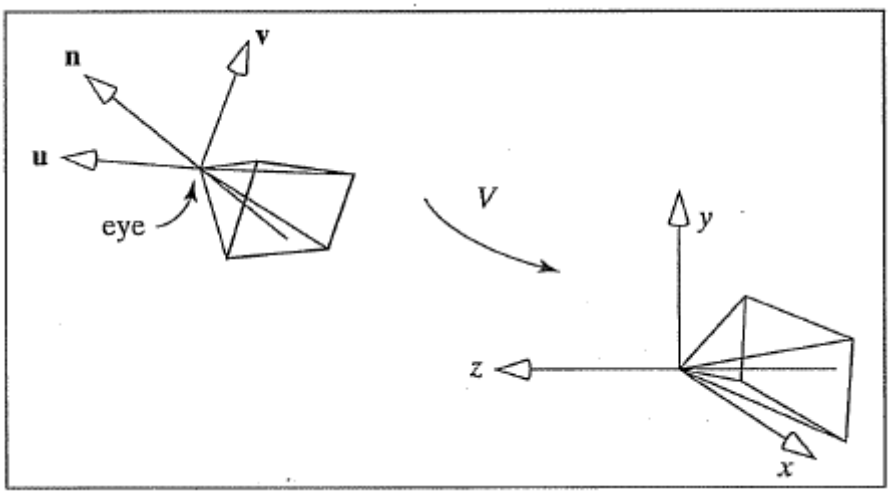
$$eye = (-2, 2, 0), look = (0, 0, 0), \text{ and } \mathbf{up} = (0, 1, 0).$$

$$\mathbf{v} = (4, 4, 0). \ \mathbf{n} = (-2, 2, 0), \mathbf{u} = (0, 0, 2), \text{ and}$$

و

$$eye = (2, 2, 0), look = (0, 0, 0), \text{ and } \mathbf{up} = (0, 0, 1).$$

$$\mathbf{u} = (-2, 2, 0) \text{ and } \mathbf{v} = (0, 0, 8).$$



شکل ۹ تبدیلي که gluLookAt() ایجاد مي کند.

در واقع ما علاقمند هستیم که تبدیل ${}^{C}_{W}T$ را بیابیم که دستگاه مختصات جهان را به دستگاه مختصات دوربین تبدیل می کند. مبدأ دستگاه جهان نسبت به دستگاه دوربین به میزان eye– جابجائی دارد که برای بیان این بردار در دستگاه دوربین باید آنرا بر روی محورهای دستگاه دوربین تصویر نمائیم. در این حالت بصورت زیر مکان یک نقطه در مختصات جهان در مختصات دوربین بدست خوهد آمد.

$${}^{C}P = {}^{C}_{W}T.{}^{W}P$$

ماتریس V که در واقع تبدیل دستگاه مختصات جهان نسبت به دستگاه دوربین $({}^{C}_{W}T)$ را بیان می دارد توسط ماتریس زیر قابل دستیابی است. اگر دقت شودخواص ماتریسهای عمود خاص در این ماتریس دیده می شود.

$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$(d_x, d_y, d_z) = (-eye \cdot \mathbf{u}, -eye \cdot \mathbf{v}, -eye \cdot \mathbf{n}).$$

برای یافتن میزان جابجائی باید مؤلفه های بردار eye– را بر روی محورهای u، v، و n بدست آوریم. برای محاسبه تصویر eye– بر روی هر یک از محورهای دستگاه دوربین کافی است آنرا در هر یک از این بردارها ضرب داخلی کنیم. برای محاسبه ستون اول ماتریس V باید تصویر بردار i را بر روی هر یک از محورهای دستگاه مختصات دوربین بدست آوریم که برای این کار مجدداً لازم است بردار $i=(1,0,0)^T$ را در هر یک از محورهای دستگاه مختصات دوربین ضرب داخلی کنیم. ستون دوم ماتریس V برابر تصویر بردار j روی محورهای دستگاه مختصات دوربین و ستون سوم ماتریس V تصویر بردار k روی محورهای دستگاه مختصات دوربین می باشد.

جهت بررسی صحت بردار V می توانیم آزمایشهای زیر را انجام دهیم.

$$V\begin{pmatrix} eye_x \\ eye_y \\ eye_z \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$V\begin{pmatrix} u_x \\ u_y \\ u_z \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

مشخص کردن دوربین در برنامه

تعریف کلاس Camera

```
cam.set(eye, look, up);    // initialize the camera - similar to
                                gluLookAt()
cam.slide(-1,0,-2); // slide the camera forward and to the left
cam.roll(30); // roll it through 30° CCW as seen by the pilot
cam.yaw(20); // yaw it through 20° to the left
cam.pitch (15); // pitch it up through 15°
etc.
```

```
class Camera{
  private:
      Point3 eye, look, up;
      Vector3 u, v, n;
      double viewAngle, aspect, nearDist, farDist; // view volume shape
      void setModelviewMatrix(); // tell OpenGL where the camera is

  public:
      Camera(); // constructor
      void set(Point3 eye, Point3 look, Vector3 up); // like gluLookAt()
      void roll(float angle); // roll it
      void pitch(float angle); // increase pitch
      void yaw(float angle); // yaw it
      void slide(float delU, float delV, float delN); // slide it
      void setShape(float vAng, float asp, float nearD, float farD);
      void getShape(float &vAng, float &asp, float &nearD, float &farD);
};
```

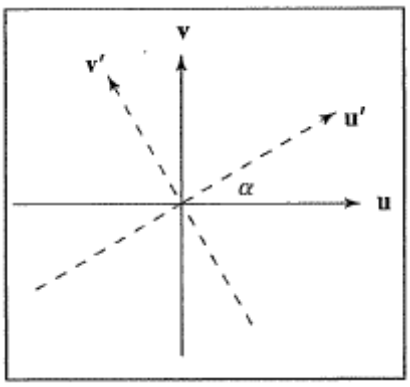
```
void Camera :: setModelviewMatrix(void)
{ // load modelview matrix with existing camera values
      float m[16];
      Vector3 eVec(eye.x, eye.y, eye.z); // constructor of a vector version of eye
      m[0] = u.x; m[4] = u.y; m[8] = u.z; m[12] = -eVec.dot(u);
      m[1] = v.x; m[5] = v.y; m[9] = v.z; m[13] = -eVec.dot(v);
      m[2] = n.x; m[6] = n.y; m[10] = n.z;m[14] = -eVec.dot(n);
      m[3] = 0; m[7] = 0; m[11]     = 0;   m[15] = 1.0;
      glMatrixMode(GL_MODELVIEW);
      glLoadMatrixf(m); // load OpenGL's modelview matrix
}
void Camera:: set(Point3 Eye, Point3 look, Vector3 up)
{ // create a modelview matrix and send it to OpenGL
      eye.set(Eye); // store the given eye position
      n.set(eye.x - look.x, eye.y - look.y, eye.z - look.z); // make n
      u.set(up.cross(n)); // make u = up X n
      n.normalize(); u.normalize(); // make them unit length
      v.set(n.cross(u)); // make v = n X u
      setModelViewMatrix(); // tell OpenGL
}
```

جابجا کردن دوربین در راستای یکی از محورهایش

حرکت در راستای u یعنی به چپ و راست رفتن، حرکت در راستای v یعنی به پائین و بالا رفتن، و حرکت در راستای n یعنی به جلو و عقب رفتن دوربین. برای حرکت دوربین به اندازهٔ D واحد در راستایu، eye را به eye+Du می نشانیم. بطور کلی برای حرکت دوربین به اندازهٔ Δu واحد در راستای u، Δv واحد در راستای v، و Δn واحد در راستای n، بردار eye را به $eye+\Delta u.u+\Delta v.v+\Delta n.n$ می نشانیم. برای آنکه دوربین دوران پیدا نکند باید مکانی را که به آن نگاه می کرده را نیز به همین اندازه جابجا نمائیم.

```
void Camera::slide(GLdouble deltaU, GLdouble deltaV, GLouble
                                deltaN){
     eye.x += deltaU*u.x + deltaV*v.x + deltaN*n.x;
     eye.y += deltaU*u.y + deltaV*v.y + deltaN*n.y;
     eye.z += deltaU*u.z + deltaV*v.z + deltaN*n.z;
     look.x += deltaU*u.x + deltaV*v.x + deltaN*n.x;
     look.y += deltaU*u.y + deltaV*v.y + deltaN*n.y;
     look.z += deltaU*u.z + deltaV*v.z + deltaN*n.z;
     setModelViewMatrix();
}
```

یافتن دوران دوربین حول محور n (roll)



$$\mathbf{u}' = \cos(\alpha)\mathbf{u} + \sin(\alpha)\mathbf{v}$$
$$\mathbf{v}' = -\sin(\alpha)\mathbf{u} + \cos(\alpha)\mathbf{v}$$

```
void Camera::roll(GLdouble angle){
GLdouble PI = 3.141592654;
//تبدیل درجه به رادیان
GLdouble radangle = PI*angle/180.0;
GLdouble cs, sn;

cs = cos(radangle);
sn = sin(radangle);
Vector3 t(u.x, u.y, u.z);    //keep original u
u.set(t.x*cs+v.x*sn, t.y*cs+v.y*sn, t.z*cs+v.z*sn);
v.set(-t.x*sn+v.x*cs, -t.y*sn+v.y*cs, -t.z*sn+v.z*cs);
setModelViewMatrix();
```

```
// the usual includes
#include "camera.h"

Camera cam; // make a global camera object

//<<<<<<<<<<<<<<<<<<<<<<<< myKeyboard >>>>>>>>>>>>>>>>>>>>>>>
void myKeyboard(unsigned char key, int x, int y)
{
  switch(key)
  {
      // controls for camera
      case 'F':    cam.slide(0,0, 0.2); break; // slide camera forward
      case 'F'-64: cam.slide(0,0,-0.2); break; //slide camera back
      // add up/down and left/right controls
      case 'P':      cam.pitch(-1.0); break;
      case 'P' - 64: cam.pitch( 1.0); break;
      // add yaw controls
      case 'Y':      cam.yaw(-1.0); break;
      case 'Y' - 64: cam.yaw( 1.0); break;
      // add roll controls
      case 'R':      cam.roll(1.0); break;
      case 'R' - 64: cam.roll( 1.0); break;
}
      glutPostRedisplay(); // draws it again
}
//<<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>>>>>>>>>
void myDisplay(void)
{
      glClear(GL_COLOR_BUFFER_BIT||GL_DEPTH_BUFFER_BIT);
      glutWireTeapot(1.0); // draw the teapot
      glFlush();
      glutSwapBuffers(); // display the screen just made
}
//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
void main(int argc, char **argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); // double buffering
      glutInitWindowSize(640,480);
      glutInitWindowPosition(50, 50);
      glutCreateWindow("fly a camera around a teapot");
      glutKeyboardFunc(myKeyboard);
      glutDisplayFunc(myDisplay);
      glClearColor(0.0f,1.0f,0.0f,0.0f);  // background is green
      glColor3f(0.0f,0.0f,0.0f);    // set color of stuff
      glViewport(0, 0, 640, 480);
      cam.set(4, 4, 4, 0, 0, 0, 0, 1, 0); // make the initial camera
      cam.setShape(30.0f, 64.0f/48.0f, 0.5f, 50.0f);
      glutMainLoop();
}
```

تمرین

۱- روالهای pitch() و yaw() در کلاس Camera را پیاده سازی نمائید.

۲- کاری کنید که بتوان توسط دکمه های موش بتوان دوربین را در محیط به حرکت در آورد.