Display the different shapes:

```
void display() {
  glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color

  // Define shapes enclosed within a pair of glBegin and glEnd
  glBegin(GL_QUADS);              // Each set of 4 vertices form a quad
    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(-0.8f, 0.1f);     // Define vertices in counter-clockwise (CCW) order
    glVertex2f(-0.2f, 0.1f);     //  so that the normal (front-face) is facing you
    glVertex2f(-0.2f, 0.7f);
    glVertex2f(-0.8f, 0.7f);

    glColor3f(0.0f, 1.0f, 0.0f); // Green
    glVertex2f(-0.7f, -0.6f);
    glVertex2f(-0.1f, -0.6f);
    glVertex2f(-0.1f,  0.0f);
    glVertex2f(-0.7f,  0.0f);

    glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
    glVertex2f(-0.9f, -0.7f);
    glColor3f(1.0f, 1.0f, 1.0f); // White
    glVertex2f(-0.5f, -0.7f);
    glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray
    glVertex2f(-0.5f, -0.3f);
    glColor3f(1.0f, 1.0f, 1.0f); // White
    glVertex2f(-0.9f, -0.3f);
  glEnd();

  glBegin(GL_TRIANGLES);          // Each set of 3 vertices form a triangle
    glColor3f(0.0f, 0.0f, 1.0f); // Blue
    glVertex2f(0.1f, -0.6f);
    glVertex2f(0.7f, -0.6f);
    glVertex2f(0.4f, -0.1f);

    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(0.3f, -0.4f);
    glColor3f(0.0f, 1.0f, 0.0f); // Green
    glVertex2f(0.9f, -0.4f);
    glColor3f(0.0f, 0.0f, 1.0f); // Blue
    glVertex2f(0.6f, -0.9f);
  glEnd();

  glBegin(GL_POLYGON);            // These vertices form a closed polygon
    glColor3f(1.0f, 1.0f, 0.0f); // Yellow
    glVertex2f(0.4f, 0.2f);
    glVertex2f(0.6f, 0.2f);
    glVertex2f(0.7f, 0.4f);
    glVertex2f(0.6f, 0.6f);
    glVertex2f(0.4f, 0.6f);
    glVertex2f(0.3f, 0.4f);
```

```
  glEnd();

  glFlush();  // Render now
}
```

window re-size event:---Called back when the window first appears and whenever the window is re-sized with its new width and height

```
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
  // Compute aspect ratio of the new window
  if (height == 0) height = 1;                // To prevent divide by 0
  GLfloat aspect = (GLfloat)width / (GLfloat)height;

  // Set the viewport to cover the new window
  glViewport(0, 0, width, height);

  // Set the aspect ratio of the clipping area to match the viewport
  glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
  glLoadIdentity();             // Reset the projection matrix
  if (width >= height) {
    // aspect >= 1, set the height from -1 to 1, with larger width
     gluOrtho2D(-1.0 * aspect, 1.0 * aspect, -1.0, 1.0);
  } else {
     // aspect < 1, set the width to -1 to 1, with larger height
    gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);
  }
}
```

Run the application

```
 main(int argc, char** argv) {
  glutInit(&argc, argv);          // Initialize GLUT
  glutInitWindowSize(640, 480);   // Set the window's initial width & height - non-square
  glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
  glutCreateWindow("Viewport Transform");  // Create window with the given title
  glutDisplayFunc(display);       // Register callback handler for window re-paint event
  glutReshapeFunc(reshape);       // Register callback handler for window re-size event
  initGL();                       // Our own OpenGL initialization
  glutMainLoop();                 // Enter the infinite event-processing loop
  return 0;
}
```

2.

Position Initialization

```
myInit()
        {
        glMatrixMode(GL_PROJECTION); // set the view volume shape
        glLoadIdentity();
                //  glOrtho(left, right, bottom, top, near, far)
        glOrtho(-5.0, 5.0, -5.0, 5.0,  -0.1,  100);

        glMatrixMode(GL_MODELVIEW); // position and aim the camera
        glLoadIdentity();
        gluLookAt(2.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);


        glClear(GL_COLOR_BUFFER_BIT); // clear the screen
      glColor3d(0,0,0); // draw lines
}


void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f (0.0, 0.0, 0.0);
        glPushMatrix();
        glTranslated(0.5, 0.5, 0.5);
        glutWireCube(6.0);
        glPopMatrix();
        glFlush();

}
```

Keyboard functions

```
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
  case 'x':
  case 'X':
    glRotatef(30.,1.0,0.0,0.0);
    glutPostRedisplay();
    break;
  case 'y':
  case 'Y':
    glRotatef(30.,0.0,1.0,0.0);
    glutPostRedisplay();
    break;
  case 'i':
  case 'I':
    glLoadIdentity();
    gluLookAt(1, 1, 1, 0, 0, 0, 0, 1, 0);
    glutPostRedisplay();
```

```
    break;
  case 's':
  case 'S':
          glScaled(0.5,0.5,0.5);
    glutPostRedisplay();
    break;

  case 27:
    exit(0);
    break;
  }
}
```

MainCode to application

```
 main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(640,480);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  glViewport(0, 0, 640, 480);
  init ();
  myInit();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```