


Exercise 02561-02: Projections – Virtual Camera

Reading	ANG: 1.4, 2.6, 4.1 - 4.4.2
Purpose	The purpose of this exercise is to understand the various method of setting up the virtual camera and to be able to adjust the parameters of the camera. We will get more acquainted with defining the matrices in the viewing pipeline and to concatenate them into the viewing matrix. Secondly, we will make different pictures of the scene using various projection method based on central projection (Front, X and 3-point perspective) and Orthographic parallel projection (Isometric, Dimetric, and Trimetric axonometric). Many of the principles will be demonstrated in OpenGL.
Part 1 Hello 3D	The program 02561-02-01 shows a 3D view of a wireframe teapot. <ul style="list-style-type: none"> Explain how the <code>Vertex</code> struct and the <code>display()</code> function differs from the ones in Exercise01.
Part 2 Model transformations in 3D	The program 02561-02-02 just makes an orthographic view of a unit cube, using <code>drawWireUnitCube()</code> . We have also included the world coordinate axis X_w , Y_w , and Z_w , and the auxiliary line through (1,0,0) and (1,3,0). The unit cube is defined in its own Object Coordinate system (O_o , X_o , Y_o , Z_o) by the diagonal vertices (0.0,0.0,0.0) and (1.0, 1.0, 1.0). The coordinate system is right-handed and the Y_o -axis points upwards. <ol style="list-style-type: none"> Scale the object to double size and rotated it 30-degree counter clockwise around the Y_w-axis. Finally, translate the object is along the Y_w-axis by three units. Which transformations did you use? In which order did you specify the transformations. Play with changing the order of the transformations.
Part 3 View transformations	In this part we shall make a perspective picture of a unit cube with the diagonal vertices (-0.5,-0.5,-0.5) and (0.5, 0.5, 0.5). <ol style="list-style-type: none"> Make a front perspective view of the cube. The eye-point should be positioned in (20, 5, 5). In a front perspective the View plane and a set of faces are parallel to each other. <ol style="list-style-type: none"> Use perspective projection using field-of-view-y set to 45 degrees. The rest of the parameters should be set to appropriate values (pay special attention to <code>zNear</code> and <code>zFar</code>). Note that you have to setup the <code>modelView</code> (next step) to see the cube after changing the projection. Make the front perspective view mentioned with the eye point (20, 5, 5) and the up = (0,1,0). Find the at-point yourself. Use the <code>LookAt</code> function to change the <code>modelView</code> matrix. Make an X-perspective from the same eye point by changing only the at-point. Play with the parameters of the two function mentioned above.
Part 4 View transformation	A cube is placed in the World Coordinate System (O_w , X_w , Y_w , Z_w) with a diagonal through the vertices (-0.5,-0.5,-0.5) and (0.5, 0.5, 0.5). We want to make an isometric view (based on orthographic parallel projection) of the cube. Use (0,0,0) as the Point of Interest (At-point) and the Z_w -axis in defining the Up-vector. Make an appropriate selection for the “eye-point”. Modify the program 02561-02-04 to create the isometric view. Use the function <code>LookAt</code> in the program.

Exercise 02561-02: Projections – Virtual Camera

<p>Part 5 LookAt in details</p> 	<p>The function <code>lookAt</code> automatically sets up the (concatenated) matrix of the viewing transformation. The viewing transformation uses a series of transformations to go from the World Coordinate System to the Eye Coordinate System.</p> <p>Exercise 02-05 shows a simple scene where you can change camera position using the keys '1' to '6'. Each key-press calls the function <code>keyN</code>, which updates the view matrix and requests a repaint. Note that view is the same for key 1 and 2 and for key 3 and 4, etc.</p> <ul style="list-style-type: none"> • Change function <code>key2</code> to use a series of view transformations instead of the <code>lookAt</code> function (without changing the view – key) • Change function <code>key4</code> to use the <code>lookAt</code> function instead of a series for translations and rotations (without changing the view) • Change function <code>key6</code> to use a hardcoded matrix (using the <code>mat4-ctor</code>) instead of the <code>lookAt</code> function.
<p>Part 6 View transformations</p>	<p>Find the viewing transformation of part 1 and 2. Find the viewing matrix - it may include finding a series of matrix transformations (Translate, Rotate, Scale).</p>
<p>Part 7 (optional)</p>	<p>Optional</p> <p>Make in OpenGL a highly oversimplified aircraft in which the body, the wings, and the horizontal and vertical stabilizers each consist of a box. Use the transformation functions <code>translate</code>, <code>rotate</code> and <code>scale</code> to position, orientate, and scale the boxes in an appropriate way. The link</p> <p>http://www.grc.nasa.gov/WWW/K-12/airplane/airplane.html</p> <p>defines term used for an aircraft and shows a detailed picture.</p> <p>Add to the wings two aileron (simplified as boxes) which you can rotate around an edge (in order to get the aircraft to roll).</p> <p>Optional you may also add boxes to simulate elevators which can change pitch (up and down) and rudders which can change yaw (side to side).</p>