Computer Graphics

Aristotle University of Thessaloniki,
Faculty of Engineering

Assignment 2: Transformation and Projection

Konstantinos Chatziantoniou
8941
konstantic@ece.auth.gr

May 15, 2020

1 Code Structure

Demos:

- demo.m Script that demonstrates a series of transformations on a given set of points and then the projection and painting of the triangles after each transformation.

Functions:

- rotationMatrix.m Returns the rotation matrix for a given angle and axis using the Rodrigues' formula.
- affineTransform.m Applies rotation and translation to a set of points and returns them.
- systemTransform.m Finds the coordinates of the points for a new coordinate system, given the axis coordinates and translation of that system.
- projectCamera.m Given the coordinates and axis of the camera, transforms the points to the camera coordinate system and finds their projection to the x-y axis and their depth.
- projectCameraKu.m Given the target's position and up vector for the camera calculates the x-y-z axis for the camera. Then calls projectCamera for the actual projection.
- rasterize.m Given the resolution of the camera and the projected points, puts the points to the grid.

- photographObject Combines the projectCamera and rasterize functions.
- objectPainter.m Function from the previous assignment, used to color the triangles formed by the projected set of points.

The demo.m script, applies 3 transformation, projects the points and paints them after each transformation. The resulting images are saved as .jpg files. There are 4 images in total, including the original pose. The *hw2.mat* has to be in the Matlab Path. Also, by setting the flag *show_figs* to *true*, the images captured by the camera and the position of the points in 3d space(scatter3) will be shown as figures.

Information about the input and ouput of the functions can be found as comments in each file.

2 Implementation Explanation

All of the functions implement simple linear algebra operations, taken from the notes, to apply transformations. The only thing different from the notes, is that *projectCamera* uses the negative *up vector*. It only changes is the oriantation of the image. This had to be different in order to agree with the indexing assumptions of the functions from the previous assignement(*objectPainter*) and the matlab's *imshow()* function.

systemTransform uses the *affineTransformFunction* for the actual transformation. *projectCamera* uses the *systemTransform* *systemTransform* and *projectCameraKu* uses the *projectCamera* for the projection and transformations.

3 Assumptions

For the *systemTransform* function, we have as input the coordinates of the axis for the new system and its translation vector from the origin. In order to apply the transform, we need the $\mathcal{L}$ affine transformation that produces the new system $\mathcal{L}\{s_{old}\} = s_{new}$. The only way to find $\mathcal{L}$ with the given arguments is to **assume** that the s_{old} axes have coordinates $x = [100], y = [010], z = [001]$.

Also, the assignment asks to rotate the points ϕ rads around a vector g that passes through the point K. The point K is not specified, so we assume the $K = [0, 0, 0]$.

4 Results

The demo was executed on an i3 4170 @ 3.7Ghz. The tranformation, projection and rasterization took aproximately **0.0015 sec** and the painting of the object took **0.8 sec** each time.
The first four figures(1-4) show the points in 3d space, along with the camera orientation and a useful vector about the next tranformation.
The final four figures (4-8), show the image captured by the camera.

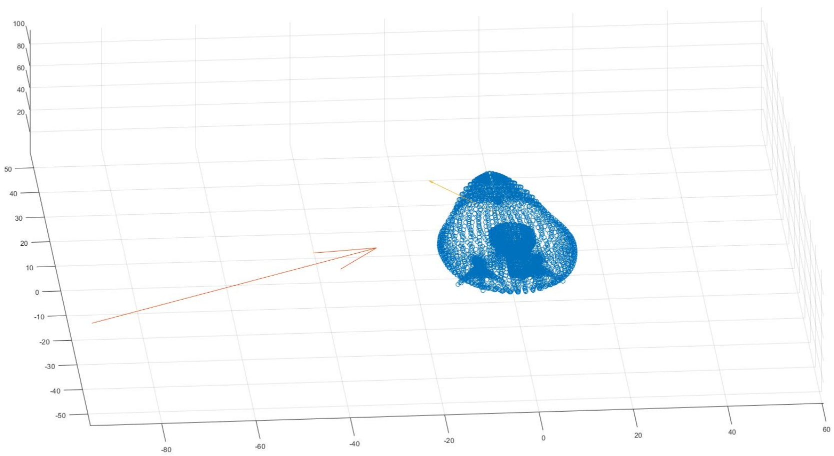

Figure 1: The duck before the translation in the direction of yellow arrow. Red arrow for the direction of z axis of Camera.

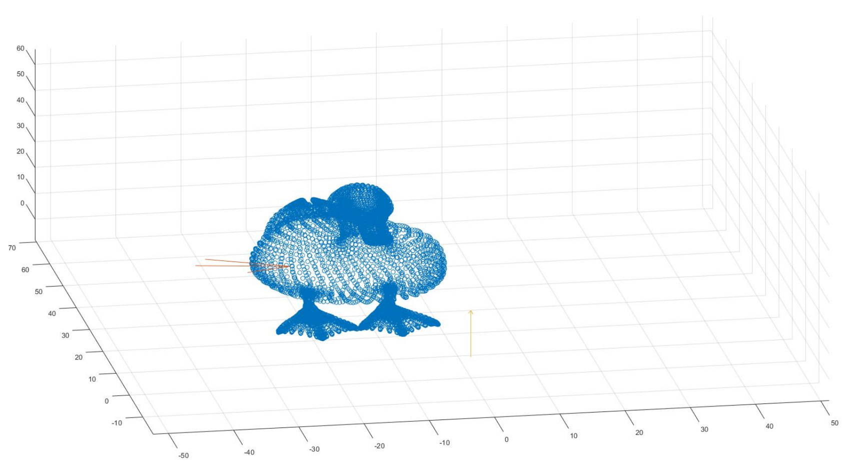

Figure 2: The duck before the rotation around the yellow arrow. Red arrow for the direction of z axis of Camera.

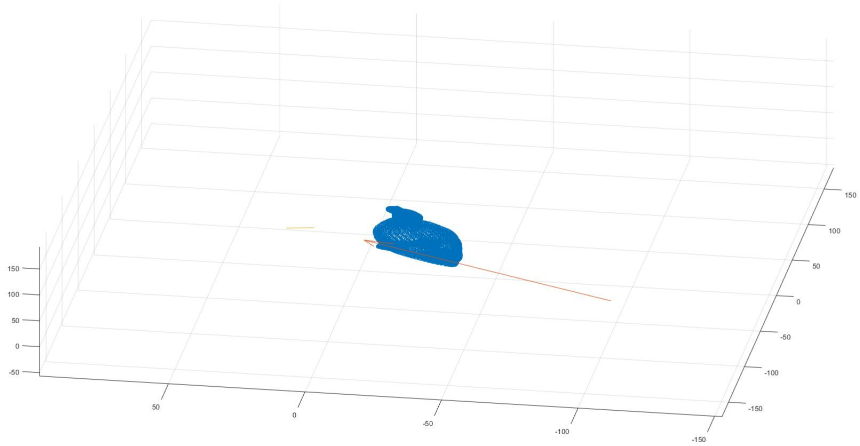

Figure 3: The duck before the translation in the direction of yellow arrow. Red arrow for the direction of z axis of Camera.

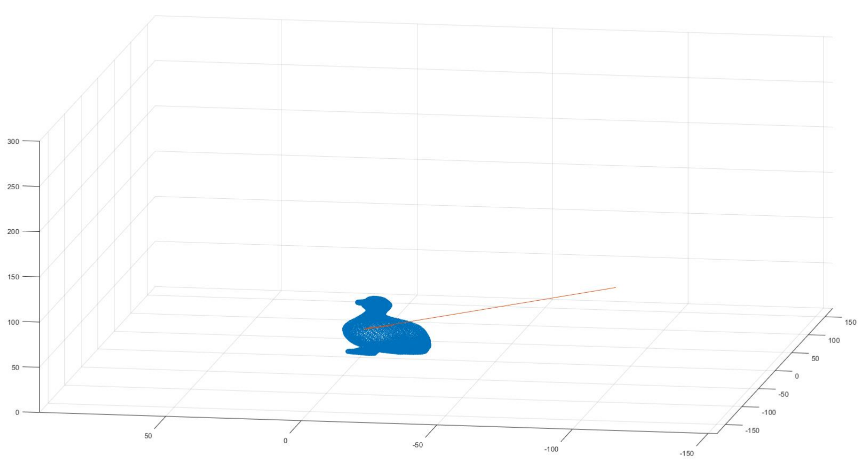

Figure 4: The final position of the duck. Red arrow for the direction of z axis of Camera.

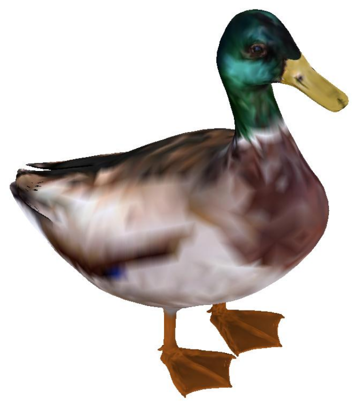

Figure 5: The initial image of the duck.

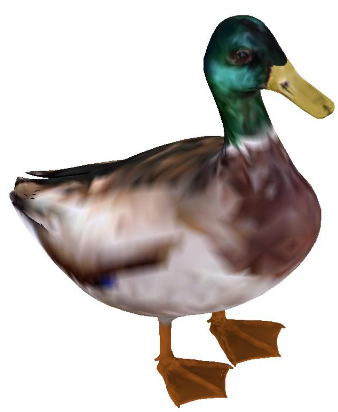

Figure 6: Image of the duck after a translation from the initial pose.

Figure 7: Image of the duck after translation + rotation from the initial pose.

Figure 8: Image of the duck after translation + rotation + translation from the initial pose.