Projections and Transformations

Computer Graphics, ECE AUTH 2022

Dimitrios Folas Demiris, AEM: 9415

1. Introduction

This project revolves around the transformation of an object in a 3-dimensional space when its or the camera's geometry changes, either they move or rotate, and the object's projection in the 2-dimensional space of a rendered image.

The project is coded in Python. As a foundation, a large part of the code from the previous part of the project was used, mainly concerning the steps needed to convert the raw. npy data into an exportable image. We are given 3 matrices: verts3d: containing 3D coordinates, vcolors: containing RGB values for every vertex, and faces: containing the index of each triangle's vertices, as well as the coordinates and the angle of the camera. Down for implementation is to translate, rotate and render each instance of the object's transformations.

2. Implementation

We utilize the Affine Transformation, which is described by the formula, to depict the movement or rotation of an object in 3D space.

$$cq = T\,cp\ (1)\,,$$

where $cp = [px\ py\ pz\ 1]^T$ is the augmented coordinate vector of the initial point p,$cq = [qx\ qy\ qz\ 1]^T$ the augmented coordinate vector of the transformed point q and T the transformation matrix.

A *transformation matrix* was implemented as a class with a 4X4 matrix T as a variable. There are two methods in it: rotate and translate. The first one builds the matrix T to rotate the vector cp by a certain angle around a given vector u, while the second one builds the matrix T to move the vector cp by a given vector t.

We also must account for the discrepancy between the two WCS, World Coordinate System and the camera coordinate system which is implemented through the systemTransform function.

Given the coordinates of the object in relation to the camera's coordinate system, next comes the application of the Perspective Projection rules to the 3D object and obtaining the projected 2D coordinates in order to produce an image. After we have the respective coordinates as projected onto the camera, we must calculate the coordinates of the frame of the image. We achieve that in the rasterize function of the code.

As previously stated to render the final images the code from the previous part is used, specifically, the smooth / gouraud rendering.

The 2nd transformation is based on the Rodriguez Rotation Formula and uses the Rodriguez Rotation matrix, implemented in the transformation class.

3. Results

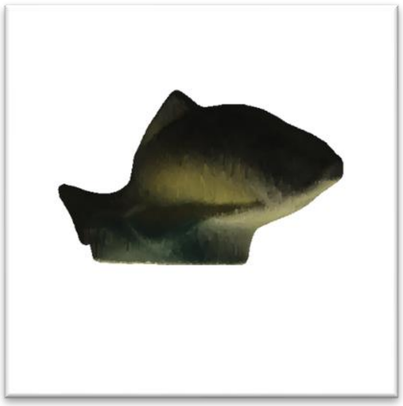
Figure 1 The original image without any transformation

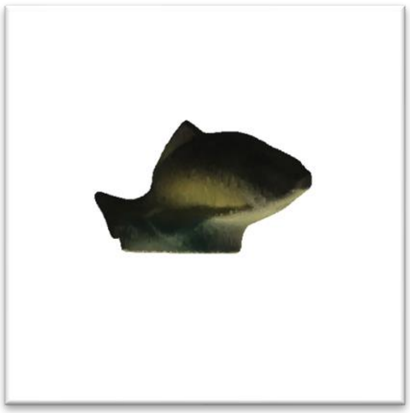
Figure 2 The image rendered after the 1st Transformation

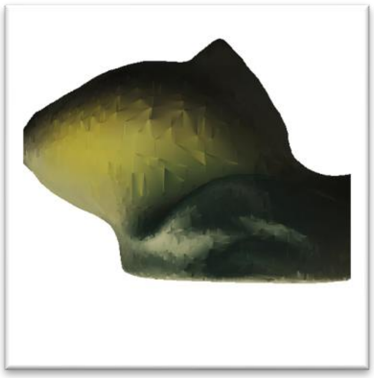
Figure 3 The image after the 2nd transformation, Rotation

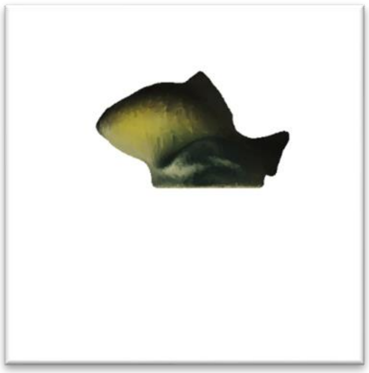
Figure 4 The image after the 4th and final transformation