Assignment 3: Viewing Matrix using General Rotation Method

Topics

- Build the world to camera matrix using the general rotation method
- Create a third person camera, first person camera and a fixed camera with COP(0,0,0,1) and View(0,0,-1,0)
- The first and third person cameras should follow the tank
- Build the camera to projection matrix using the perspective transform explained in class
- Transform the projected points to NDC space (-1 to 1)
- Update the NDC to viewport matrix accordingly
- Populate the world with cubes to help see better the tank navigation

Description

Given an input file "input.txt" containing the following information:

- The camera/projection window left, right, top, bottom, focal, near plane and far plane values (ignore the near and far values for this assignment)
- A vertex list defining a cube centered at the origin and of size 1 (defined in model space)
- A face list holding the indexes to the triangles in the cube
- A face color list holding the colors of the given faces
- A texture coordinates list holding the texture coordinates of all vertexes in the cube (number of texture coordinates is equal to number of faces * 3)

The application should:

1. Read the input file
2. Accept from the keyboard the user input in order to navigate an object (the Tank)
3. Compute the camera position and target.
4. Build the camera transformation according to the user input
5. Allow to swap between first and third person camera
6. Populate the world with a few cubes
7. Cull in camera space using the naïve culling method (compare z-values farther than near)
8. Project the vertexes
9. Map the vertexes from the projection plane to the view-port
10. Rasterize on the screen

Camera

Camera properties should also be extracted from the file. Left/Right give information about the width of the window and same with Top/Bottom for the height. For this assignment *far* and *near* data should be added to the perspective matrix.

When creating the view matrix, make sure that the view vector gets aligned with the **negative Z axis** and that your perspective matrix also takes that into account.

Tank

Our simple tank is composed of: a body with a turret and four wheels directly attached to it; a gun, which is attached to the turret. Based on all these pieces, there are three vectors that we will need to move the tank and to place and orient our first and third-person cameras. These vectors are: the forward vector, the view vector and the up vector. All these three vectors need to be expressed in the same coordinate system (the world's).

- **Forward Vector:** As shown in Figure 1, the forward vector represents the direction of motion of the tank, and it is its body's Z axis.
- **View Direction:** As shown in Figure 1, the view vector represents the direction of view of somebody sitting on the turret, and it is the turret's Z axis.
- **Up Direction:** In general, the up direction of the tank is the aligned with the body's Y axis. (note: for this simple programming assignment, this should always match the world's Y axis).

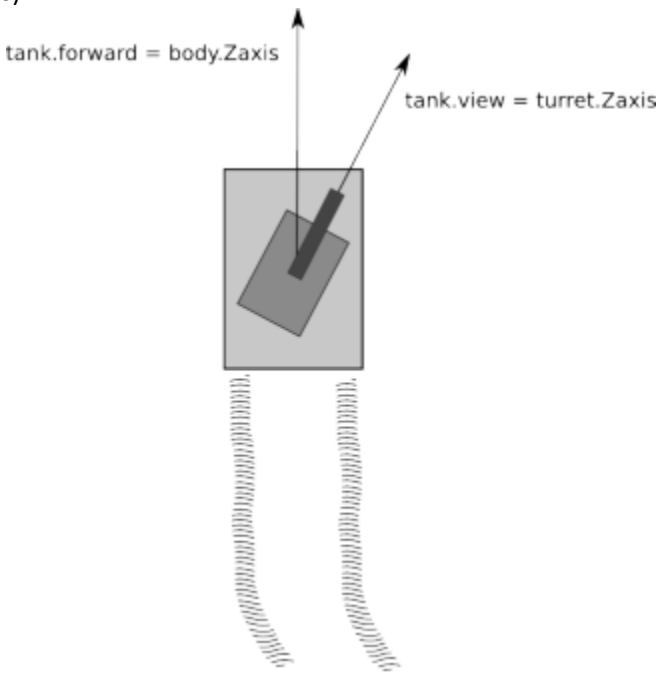

Figure 1

First-Person Camera

A first-person camera (for this tank exercise) should be positioned at the center of the turret. The camera's view vector should be aligned with the turret's Z axis. The camera's up vector will be aligned with the tank's up vector. The camera's right vector will computed to complete a left-handed coordinate system.

$$\begin{cases} camera.position = turret.position \\ \overrightarrow{camera.view} = \overrightarrow{turret.Zaxis} \\ \overrightarrow{camera.up} = \overrightarrow{turret.Yaxis} \\ \overrightarrow{camera.right} = \overrightarrow{camera.view} \times \overrightarrow{camera.up} \end{cases}$$

Third-Person Camera

We are going to introduce two scalar values to help us control the position and orientation of the third-person camera respect to the tank:

- Distance: represents how far behind along the horizontal (XZ plane) the camera is from the tank.
- Height: represents how high the camera is respect to the tank (see figure 2).

$$\begin{cases} \overrightarrow{tank.forward} = \overrightarrow{body.Zaxis} \\ \overrightarrow{tank.up} = \overrightarrow{body.Yaxis} \end{cases}$$

The position of the camera (i.e. the center of projection) is calculated as:

$$camera.position = tank.position - distance \cdot \overrightarrow{tank.forward} + height \cdot \overrightarrow{tank.up}$$

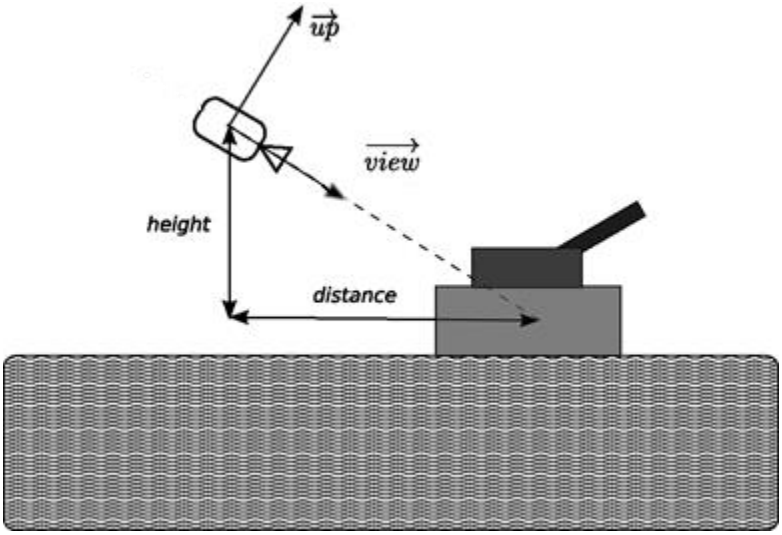

Figure 2

The camera's view vector is computed as:

$$\overrightarrow{camera.view} = \frac{tank.position - camera.position}{|tank.position - camera.position|}$$

We compute the camera's right vector next:

$$\overrightarrow{camera.right} = \overrightarrow{camera.view} \times \overrightarrow{camera.up}$$

And, finally, we compute the camera's up vector to complete a left-handed coordinate system:

$$\overrightarrow{camera.up} = \overrightarrow{camera.right} \times \overrightarrow{camera.view}$$

Scene

Along with the tank from Assignment 2 the scene needs to include a grid of cubes of scale (10,10,10) throughout the XZ-plane. Add a 5x5 grid of cubes from positions (-100,0,-100) to (100,0,100).

Input

- 1: wireframe mode
- 2: solid mode
- 3: first person mode
- 4: third person mode
- 5: rooted camera mode
- a: Rotates tank body/Right rotation(ccw)
- d: Rotates tank body/Left rotation (cw)
- q: Rotates turret/Left rotation (ccw)
- e: Rotates turret/Right rotation (cw)
- r: Rotates gun/Up rotation (cw)
- f: Rotates gun/Down rotation (ccw)
- space: Move tank body forward
- z: Decrease camera distance
- x: Increase camera distance
- h: Decrease camera height
- y: Increase camera height

Grade Breakdown

Feature	Grade %
Camera position calculation	15%
View vector calculation	5%
View matrix translation	10%
View matrices rotations	25%
Projection matrix	25%
NDC to viewport matrix	10%
Code quality	10%