Assignment 2: Simple Perspective Projection & Transformation

Topics

- Build your CS250 main framework using SFML
- Implement a matrix stack functionality
- Build the model to world matrix
- Assume the world to camera matrix is the identity matrix
- Build the simple perspective matrix
- Build the projection to viewport matrix
- Transform the vertexes to viewport space
- Write a line rasterizer
- Write a triangle rasterizer
- Draw your scene in both wireframe and solid mode

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)

Using the cube vertexes database we will build a tank having the following body parts:

- Body
- Turret
- Gun
- 4 Wheels

Also

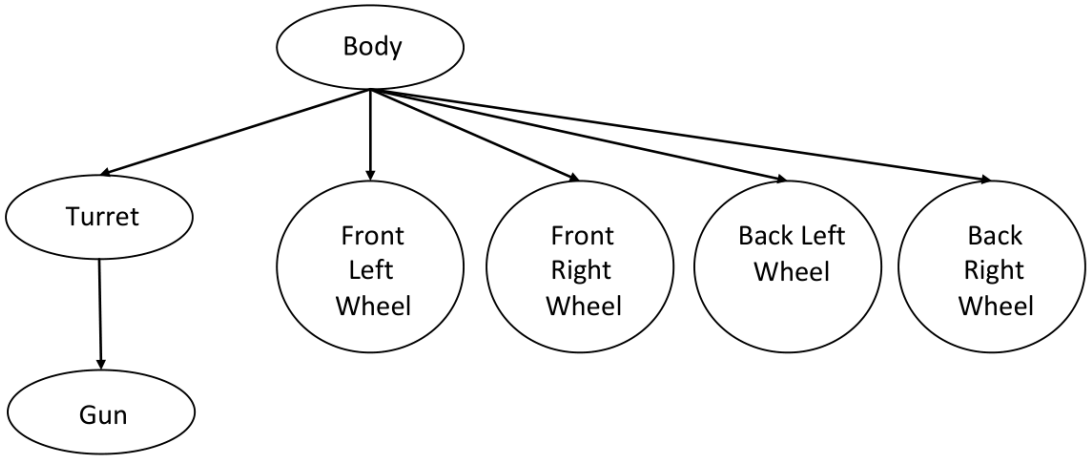
- The turret is attached to the tank body
- The gun is attached to the turret
- The wheels are attached to the tank body

The assignment is about building a transformation based on the keyboard input that transforms the tank from the model space to the world space. The tank is made from several objects that are linked. (See tank hierarchy diagram below). When the tank body is transformed (translation and rotation) the turret, gun, and wheels are affected. When the turret is rotated, the gun is affected. We will use a stack to support the hierarchy transformation. Once the vertexes are in the world space build and use the perspective matrix to project the vertexes onto the projection plane. Next, you need to transform the vertexes to the view port space and rasterize.

The application should:

1. Read the input file
2. Accept from the keyboard the user input
3. Transform the tank (made from 7 cubes) according to the user input (details provided below) to the world space
4. Project the vertexes onto the projection plane
5. Map the vertexes from the projection plane to the viewport
6. Rasterize the cube in both wireframe and solid in the viewport space

The following diagram will show the tank hierarchy:



Note: On this implementation scale is **not** propagated from parent to children transformations.

Tank information

- Body dimensions or scale values: Sx=30,Sy=25, Sz=80
- Body Initial Position: Anywhere along the Negative z-axis
- Body rotation axis: Y-axis
- Body initial angle: 0 degrees

- Turret dimensions or scale values: Sx=25, Sy=15, Sz=25
- Turret Initial position: (0, 20, 0)
- Turret rotation axis: Y-axis
- Turret initial angle: 0 degrees

- Gun dimensions or scale values: Sx=5, Sy=5, Sz=40
- Gun Initial position: (0, 0, 12.5)
- Gun rotation axis: X-axis
- Gun initial angle: 0 degrees

- Wheel dimensions or scale values: Sx=5, Sy=20, Sz=20
- Wheel 1 Initial position: (17.5, -12.5, -25)
- Wheel 2 Initial position: (-17.5, -12.5, -25)
- Wheel 3 Initial position: (17.5, -12.5, 25)

- Wheel 4 Initial position: (-17.5, -12.5, 25)
- Wheel rotation axis: X-axis
- Wheels initial angle: 0 degrees

Input

- 1: wireframe mode
- 2: solid 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

Mesh and camera

Your application should read these data from *input.txt*:

- Mesh: The only mesh used all throughout this assignment is a cube and the vertices, faces (triangles) and colors are listed on the file.
- 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 not all camera attributes are needed, so *far* and *near* data should be ignored.

Note: A parser to read all the data is given with the assignment material.

Grade Breakdown

Feature	Grade %
Body local transformation	10%
Wheel local transformations	10%
Turret local transformation	10%
Gun local transformation	10%
Children parent transform concatenations	10%
Simple perspective matrix	20%
Viewport matrix	10%
Controls/Input	5%
Code presentation	10%
Solid/Wireframe mode	5%