PA3 Part B Documentation

Program Description

Calculates the areas of two triangles given their vertices and the affine map needed to transpose the first triangle to the second triangle using the first and fourth points to translate.

Important Library Details

- Eigen
  - Library path: the headers for the Eigen library are located in /usr/include/eigen3 on my Linux machine.
  - Library version: I have installed Eigen version 3.4.0.

Marginal Cases

- Invalid inputs:
  - V is singular. Found with $det(A) = 0$.
- Invalid computations:
  - All important computations in the Eigen implementation methods were handled by Eigen, and the outputs have been checked.
  - Each output will be manually checked for validity.

Design Choices

- V is defined as $V = \left[\vec{v}_2, \vec{v}_3\right]$ where $\vec{v}_2 = \overrightarrow{a_1 a_2}$ and $\vec{v}_3 = \overrightarrow{a_1 a_3}$. Likewise, V' is defined as $V' = \left[\vec{v'}_2, \vec{v'}_3\right]$ where $\vec{v'}_2 = \overrightarrow{a'_1 a'_2}$ and $\vec{v'}_3 = \overrightarrow{a'_1 a'_3}$.
- The areas will be solved by using the determinant to calculate area of the relevant parallelogram and halving it(since determinants can be negative, the absolute value of the determinant is used):
  - $Area_V = \frac{1}{2} abs(|V|)$
  - $Area_{V'} = \frac{1}{2} abs(|V'|)$
- The relevant affine map can be calculated with $A = V'V^{-1}$.

Pseudocode

```
Matrix(2,6) GetInputAsMatrix(const string &input_path)

Int main():
        CALL SolveFile() for each file
        RETURN 0

double TriangleArea(const matrix &triangle_mat):
        RETURN 0.5 * abs(triangle.determinant())

void SolveFile(const string &input_path, const string &output_path):
        DECLARE 2x6 matrix raw_input as CALL of GetInputAsMatrix with input_path
        OPEN output_file at output_path

        CREATE V from first 3 columns of raw_input
        CREATE V’ from last 3 columns of raw_input

        RUN PartA(V, V’, output_file)
        RUN PartB(mat, output_file)

        CLOSE output_file

void PartA(const matrix &v_mat, const matrix &v_prime_mat, ofstream &output_file):
        PRINT TriangleArea(v_mat) to output_file
        PRINT TriangleArea(v_mat_prime) to output_file

void PartB(const matrix &v_mat, const matrix &v_prime_mat, ofstream &output_file):
        IF v_mat.determinant = 0:
                PRINT “Cannot compute” to output_file
        ELSE:
                DECLARE matrix map as v_prime_mat * v_mat.inverse()

                PRINT map to output_file
```