PA5 Part D Documentation

Program Description

Calculates whether a line intersects a triangle and the point the line intersects on the plane of said triangle.

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:
  - Line vector is parallel to plane($\vec{v} \bullet \vec{n} = 0$): The line never touches the plane therefore no intersection exists or infinitely many exist.
  - Triangle forms a line or a point(edges of triangle are parallel: $\vec{v}_1 \bullet \vec{v}_2 = \pm\ ||\vec{v}_1||\ ||\vec{v}_2||$): no barycentric coordinates can be generated if the triangle does not have area nor can a plane be found.(derived from the dot product theorem, where θ = 0° or 180°)
- Invalid computations:
  - All important computations in the Eigen implementation methods were handled by Eigen, and the outputs have been checked.

Design Choices

- $\mathbf{x}$ will be solved using the line-plane intersection algorithm like so: $\mathbf{x} = \mathbf{p} + \frac{(\mathbf{q}-\mathbf{p})\bullet\vec{n}}{\vec{v}\bullet\vec{n}}\vec{v}$.
- Solving if a point is in a triangle:
  - OLD SOLUTION: The barycentric coordinates of intersections on a triangle(u_1 and u_2) can be solved using Cramer's rule. The equation to be solved will be $\mathbf{x} = u_1\mathbf{p}_1 + u_2\mathbf{p}_2 + u_3\mathbf{p}_3$, re-expressed in matrix form as $\begin{bmatrix} u_1 p_{1,1} + u_2 p_{2,1} + u_3 p_{3,1} \\ u_1 p_{1,2} + u_2 p_{2,2} + u_3 p_{3,2} \\ u_1 p_{1,3} + u_2 p_{2,3} + u_3 p_{3,3} \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ p_1 & p_2 & p_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, solving only for u_1 and u_2. This is far from the most efficient solution, but it is the simplest.

Unfortunately, I encountered an edge case where a triangle defined by $\begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ that I could not figure out a bug patch for.

- NEW SOLUTION: I discovered a method outlined in a blog (https://gdbooks.gitbooks.io/3dcollisions/content/Chapter4/point_in_triangle.html) which works by adding up the angles between 2 points with the point inside as a corner and seeing if the sum is equal to 2π. This works for any point inside of the triangle and its edges: the only edge cases are the corners, which are easy to account for.

Pseudocode

```
STRUCT Input:
      MEMBER line_
      MEMBER triangles_

FUNCTION GetInput(input_path)

FUNCTION AreParallel(vec_1, vec_2):
      DECLARE dot_product = vec_1.dot(vec_2)
      DECLARE magnitude_product = vec_1.norm() * vec_2.norm()

      RETURN dot_product == magnitude_product or
             dot_product == -magnitude_product

FUNCTION AngleBetweenVecs(vec_1, vec_2):
      DECLARE dot_product = vec_1.dot(vec_2)
      DECLARE magnitude_product = vec_1.norm() * vec_2.norm()

      RETURN acos(dot_prouct / magnitude_product)

CLASS Triangle:
      MEMBER plane_
      MEMBER point_1_
      MEMBER point_2_
      MEMBER point_3_

      MEMBER has_area_
```

```
      CONSTRUCTOR(point_1, point_2, point_3):
            SET point_1_ = point_1
            SET point_2_ = point_2
            SET point_3_ = point_3

            DECLARE v_0 = point_2_ - point_1_
            DECLARE v_1 = point_3_ - point_1_

            IF AreParallel(v_0, v_1):
                  SET has_area_ = false
            ELSE:
                  SET plane_ = PointNormalPlane(v_0.cross(v_1), point_1_)

                  SET has_area_ = true

      FUNCTION IsPointInside(point):
            IF point == point_1_ or point == point_2_ or point == point_3_:
                  RETURN true
            ELSE
                  DECLARE v_1 = point_1_ - point
                  DECLARE v_2 = point_2_ - point
                  DECLARE v_3 = point_3_ - point

                  DECLARE sum = AngleBetweenVecs(v_1, v_2) +
                                AngleBetweenVecs(v_1, v_3) +
                                AngleBetweenVecs(v_2, v_3)

                  RETURN sum == 2 * π

CLASS PointNormalPlane:
      MEMBER normal_vec_
      MEMBER normal_vec_tail_

      CONSTRUCTOR PointNormalPlane(normal_vec, normal_vec_tail):
            SET normal_vec_ = normal_vec.normalized
            SET normal_vec_tail_ = normal_vec_tail

CLASS ParametricLine:
      MEMBER vec_v_
      MEMBER point_on_line_
```

```
      CONSTRUCTOR ParametricLine(point_1, point_2):
            SET vec_v_ = point_2 - point_1
            SET point_on_line_ = point_1

FUNCTION IntersectionExists(plane, line):
      RETURN plane.GetNormalVec().dot(line.vec_v_) != 0

FUNCTION main():
      CALL SolveFile() for each file

      RETURN 0

FUNCTION GetIntersection(plane, line):
      DECLARE intersection = line.vec_v_

      DECLARE numerator = plane.normal_.dot(plane.normal_tail_ -
                                            line.point_on_line_)
      DECLARE denominator = line.vec_v_.dot(plane.normal_)

      SET intersection = intersection * (numerator / denominator)

      SET intersection = intersection + line.point_on_line_

      RETURN intersection

FUNCTION SolveFile(input_path, output_path):
      DECLARE input raw_input = CALL of GetInput with input_path
      OPEN output_file at output_path

      FOR triangle in raw_input.triangles_:
            IF Triangle.HasArea and
                        IntersectionExists(triangle.plane_, raw_input.line_):
                  DECLARE intersection = GetIntersection(triangle.plane_,
                                                         raw_input.line_)

                  IF triangle.IsPointInside(intersection):
                        PRINT “1” to output_file
                  ELSE:
                        PRINT “0” to output_file

                  PRINT intersection to output_file
            ELSE:
```

```
            PRINT “Invalid Computation” to output_file

CLOSE output_file
```