Programming Assignment # Software Design Document

CS2300 Section 4 Fall 2021

Caleb Harris

Project Description:

This project with have three parts. The first part will calculate if a triangular planar is front or back facing, calculates the light intensity reflected off that object, then combine both into an approach that simulates shading an object if it is facing towards us. Part two had two sub parts. The first part computed parallel projection of a point x onto a plane. The second part computed the perspective projection of a point x onto a plane towards the origin. Lastly, part three had two sub parts as well. First, the distance from a point to a plane is computed and second, is to find if a line intersects with a triangle.

Approach

Part A

1. Get input file name
   1. If not valid, get a new name
2. Create an output file name based on input file name
3. Assign eye location = first three values in input
4. Assign Light direction = next three values
5. Ignore last 3 on first line
6. Define next three groups of 3 as points p, q, and r
7. Create a triangle object with these points
8. Store all triangles in array list.
9. Set view vector for all triangles.
10. Print the output.

Part B

1. Get input file name
   1. If not valid, get new name
2. Create an output file based on input file name
3. Get point on plane q from input, get normal vector n, and get projection direction v
4. Get all points from input file and store in array list
5. Compute all parallel projection points
6. Compute all perspective projection points
7. Output projection points into 2 different files

Parc C

1. Get input file name
   1. If not valid, get new name
2. Create an output file based on input file name
3. Create planes for each line of input
4. Assign last 3 numbers of each line to a point x stored on each plane.
5. Compute and display each distance x from defined plane.
6. Define a line object
7. See if line intersects with triangle
8. Output whether it intersections or not.

Detailed Design

Programming Language

The code is written in Java and was tested in version 10.0.22000 Build 22000 on Windows 11.

Methods

* partAComputations() -> function opens the passed in fileName completes the computations and outputs to a file.
* partBComputations() -> completes part b computations
* partCComputations() -> completes part c computations
* getInputFile() -> gets the input file name from user
* getOutputFile() -> gets an output file name based on the input file name passed in
* getUserChoice() -> get user choice returns a y or n for the user choice
* findRows() -> finds the amount of rows in the file passed in
* printOutput() -> prints output to file
* setViewVector() -> sets the view vector and normal vector for part a computations
* isCulling() -> sees if culling or not
* calcLightIntensity() -> calculates light intensity
* matrix operations methods
* computePerspectiveProjection() -> computes the perspective perspection
* computeParallelProjection() -> computes parallel projection
* fgetLine() -> gets a line from input file
* printDists() -> prints the distances
* fscanPlanes() -> reads point q, vector n, and point x from file
* isIntersecting() -> sees if line intersects triangle or not
* calcDistXFromPlane() -> calculates distance x is from plane

Flowcharts

# Part A)

Diagram

Description automatically generated

# Part B)

Diagram

Description automatically generated

# Part C)

Diagram

Description automatically generated

Key Data Structures

Part A

* P04\_Triangle

|  |  |
| --- | --- |
| Private Data Fields | Methods |
| p | setViewVector() |
| q | isCulling() |
| R | calcLightIntensity() |
| Centroid | Shade() |
| View vector | setNormalVector() |
| Normal vector | subPointsOrVectors() |
|  | calcCrossProd() |
|  | calcLength() |
|  | calcDotProd() |

Part B

* No defined data structures.
* Stored variables q, n, v, and points x in arrays.

Part C

* P04\_Plane

|  |  |
| --- | --- |
| Private Data Fields | Methods |
| Q | calcDistXFromPlane() |
| N |  |
| x |  |

* P04\_Line

|  |  |
| --- | --- |
| Private Data Fields | Methods |
| Point1 | isIntersecting() |
| Point2 |  |
| Point3 |  |