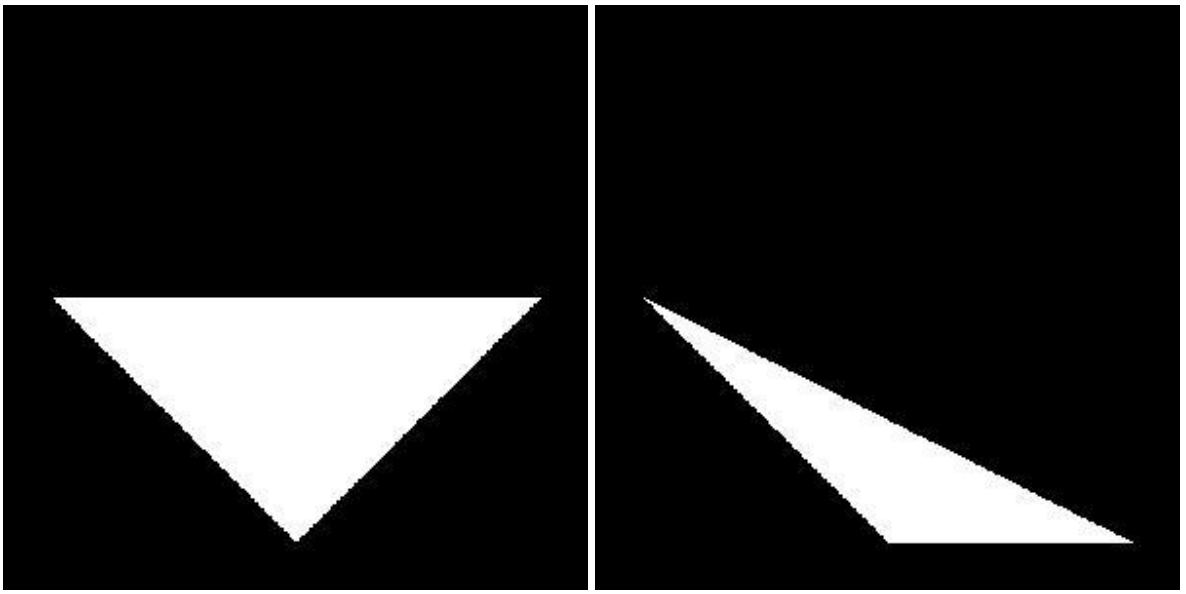
Programming Assignment
Ray Tracer



Ray Trace Triangle:

For the base of my ray-trace programme, I used PILLOW, a python image library to draw pixels to an image then display it. To do this the programme cycles through all of the pixels x,y on-screen with an offset because PILLOW automatically sets 0,0 in the top left corner yet in 3D 0,0,0 is in the screen centre. Throughout the program, it calculates the direction and origin position of the ray which is then passed into my RayTraceTriangle() function. I researched this algorithm from (www.scratchapixel.com, n.d). Within the function, a Moller Trumbore's Algorithm is used to detect an intersection with the triangle. This algorithm is based on converting the coordinates of the intercept into a barycentric coordinate which uses two edges of the triangle both with one common point to create a coordinate system. This leaves 3 values unknown u, v and t the scalar quantities in front of the 3 edge vectors that make up the coordinate system. However we only need to do the calculations for 2 as u + v + t = 1 therefore the last unknown 1 - u - v = t. The best way to calculate these unknowns is via Cramer's rule which solves for multiple unknowns in our case u, v and t. This rule states that If a matrix (MainMatric) is filled with the scaler associated with each unknown from the barycentric coordinate equation in a horizontal format, is multiplied by a column vector of the unknown values (UnkownMatic), it equals a vertical matrix of the known result. The unknown values can then be found by dividing the determinant of the MainMatric which has had column x replaced by UnkownMatic increasing x for each unknown. Finding these determinants for a 3x3 matrix is computable with a dot and cross production otherwise known as a scalar triple product. Which is the method used within my code.

[12.04096031, 2.869147062, 58.84928131, 9.723209381, 3.718580484, 61.6500473, 13.54049683, 4.058139801, 59.8635025, 14.16813087, 5.893065929, 56.33564377, 12.040960
31, 2.869147062, 58.84928131, 13.54049683, 4.058139801, 59.8635025, 14.16813087, 5.893065929, 56.33564377, 13.54049683, 4.058139801, 59.8635025, 14.91048622, 8.4351
62544, 57.58205414, 13.21987724, 10.29722309, 57.492939, 14.16813087, 5.893065929, 56.33564377, 14.91048622, 8.435162544, 57.58205414, 13.21987724, 10.29722309, 57.
492939, 14.91048622, 8.435162544, 57.58205414, 11.16208935, 10.25540447, 61.98350143, 11.67113113, 10.73329639, 61.02169037, 13.21987724, 10.29722309, 57.492939, 11
.16208935, 10.25540447, 61.98350143, 8.894145012, 7.739738464, 62.6896553, 11.67113113, 10.73329639, 61.02169037, 11.16208935, 10.25540447, 61.98350143, 10.57409191
, 4.880725861, 62.78512573, 8.894145012, 7.739738464, 62.6896553, 11.16208935, 10.25540447, 61.98350143, 9.354844093, 11.23891258, 57.13477707, 9.200230598, 11.0585
3653, 57.88923264, 8.779807091, 9.878797531, 60.13775253, 9.671468735, 1.83815074, 55.82570267, 7.295972347, 1.528506398, 58.57918549, 8.295594215, -0.0004877253959
, 57.02821732, 11.82179451, 9.413736343, 53.09732056, 11.88723183, 9.437506676, 52.90758133, 11.82179451, 9.413736343, 53.09732056, 12.96119022, 12.24084759, 59.506
5155, 13.34811211, 11.93563843, 57.14953232, 10.1465435, 12.5121851, 61.0266571, 8.929807663, 7.706118107, 64.37194061, 7.745656013, 6.826281071, 63.92277527, 9.358
390808, 2.192954063, 61.98861694, 12.74704075, 1.347223043, 56.46414566, 14.01377773, 2.25448823, 58.12911987, 15.48282528, 4.381575584, 56.57882309, 14.19726563, 7
.004294872, 54.93928146, 14.19261169, 3.268300295, 54.8754158, 15.05841255, 7.853007317, 53.87919998, 14.19726563, 7.004294872, 54.93928146, 15.05841255, 7.85300731
7, 53.87919998, 12.99842072, 12.599123, 56.34807968, 11.32573605, 12.43902683, 58.4544754, 14.19726563, 7.004294872, 54.93928146, 12.99842072, 12.599123, 56.3480796
8, 11.32573605, 12.43902683, 58.4544754, 12.99842072, 12.599123, 56.34807968, 9.868602753, 13.37872791, 60.12345886, 8.063781738, 9.740125656, 62.36706161, 11.32573
605, 12.43902683, 58.4544754, 9.868602753, 13.37872791, 60.12345886, 8.063781738, 9.740125656, 62.36706161, 9.868602753, 13.37872791, 60.12345886, 7.691216469, 11.1
8812847, 62.76687241, 7.691216469, 11.18812847, 62.76687241, 6.721391201, 7.880222797, 63.93786621, 7.276955605, 3.483456612, 63.21180725, 8.063781738, 9.740125656,
62.36706161, 7.691216469, 11.18812847, 62.76687241, 7.276955605, 3.483456612, 63.21180725, 8.172576904, 3.497331619, 62.24929428, 8.063781738, 9.740125656, 62.3670
6161, 7.276955605, 3.483456612, 63.21180725, 8.172576904, 3.497331619, 62.24929428, 7.276955605, 3.483456612, 63.21180725, 10.12672043, 0.8314138055, 59.79207993, 1
2.18650723, 1.892350197, 57.34511566, 8.172576904, 3.497331619, 62.24929428, 10.12672043, 0.8314138055, 59.79207993, 8.894145012, 7.739738464, 62.6896553, 9.7232093
81, 3.718580484, 61.6500473, 9.358390808, 2.192954063, 61.98861694, 8.894145012, 7.739738464, 62.6896553, 9.358390808, 2.192954063, 61.98861694, 7.745656013, 6.8262
81071, 63.92277527, 8.894145012, 7.739738464, 62.6896553, 7.745656013, 6.826281071, 63.92277527, 10.1465435, 12.5121851, 61.0266571, 10.35607529, 11.02180576, 60.82
844543, 8.894145012, 7.739738464, 62.6896553, 10.1465435, 12.5121851, 61.0266571, 13.21987724, 10.29722309, 57.492939, 10.35607529, 11.02180576, 60.82844543, 10.146
5435, 12.5121851, 61.0266571, 13.21987724, 10.29722309, 57.492939, 10.1465435, 12.5121851, 61.0266571, 13.34811211, 11.93563843, 57.14953232, 14.16813087, 5.8930659
29, 56.33564377, 13.21987724, 10.29722309, 57.492939, 13.34811211, 11.93563843, 57.14953232, 14.16813087, 5.893065929, 56.33564377, 13.34811211, 11.93563843, 57.149
53232, 15.16084003, 7.751297951, 54.96455383, 14.16813087, 5.893065929, 56.33564377, 15.16084003, 7.751297951, 54.96455383, 12.81322861, 1.770661116, 57.80579758, 1

File reading:

In my file reading script, I used (Stack Overflow, n.d) as a reference. I started by iterating through each line in the file, then in that line, I iterated through every space to split up the 3 sets of coordinates. Next, I iterated through every ‘,’ to separate each number which is then stored into an array called f_list. Secondly, the program then iterates through the array f_list 9 times assigning those 9 values into blocks of 3 for each vertex's coordinates. This makes up one triangle and is then stored in the scene array to be iterated through in the render() function.

Rendering complete object:

To render the full object I passed the list of triangles into the ray-tracing function. If the function returns true then a ray has been intercepted by a triangle and I converted that pixel to white. However, my program was very slow therefore I added a few functionalities to speed the program up so that I could work on correcting the shading quicker. First I pre-calculated all of the information that would be the same for all points on a triangle and assigned them to the triangle class stored within the scene array, the normal and each edge vector instead of recalculating. Secondly, I included backface culling. To do this the program uses the pre-calculated normal and compares them against the camera direction. If the dot product is less than -0.2 then the face is pointing away from the camera therefore I don't render it. This nearly halves the number of objects the ray tracer has to check.

Light:

*In terms of the shading model, I used a simple diffuse shading model which I calculated from the dot product of the normalised normal of the surface against the normalised LightPoint vector. Then multiplied by the incident lighting and the object RGB to simulate a clean drop off from the light source as well as colouring the object. The light source is represented by a simple class containing the position, RGB colour and brightness. The position is used to calculate the vector from the intercept point to the light source which is used in the diffuse and incident light calculation. Alongside incident light and the diffuse lighting, I also included ambient light to add colour to the areas that were not hit by any light at all so that they don't blend into the background. This colour was generated by 0.25 * my objects colour.*

Shadows:

To trace shadows I reused the RayTraceTriangle() and shot another ray from the point intercept going in the direction of the vector between the intercept and the light. The RayTraceTriangle() with these parameters then checks against all the objects in the scene and if any were hit by the ray on the way to the light then another object is casting a shadow on this pixel resulting in the colour value being halved. However, if the length of the ray is too short it is discarded to avoid triangles casting shadows on themselves.

References:

www.scratchapixel.com. (n.d.). Ray Tracing: Rendering a Triangle (Möller-Trumbore algorithm). [online] Available at: https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/moller-trumbore-ray-triangle-intersection.(Accessed 24 - 27 January 2022)

Stack Overflow. (n.d.). floating-point - Reading floats from file with python. [online] Available at: https://stackoverflow.com/questions/44975599/reading-floats-from-file-with-python (Accessed 24 Jan. 2022).

Final Render