The polygonal surface model of a 3D object and removing invisible surfaces

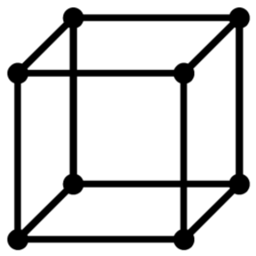
Aim: The study of the process of the construction of parametric 3D object and visualization it with removing invisible surfaces.

Task: Implement software for the construction and visualization of the polygonal surface model using parallel and perspective projections with Z-buffer algorithm.

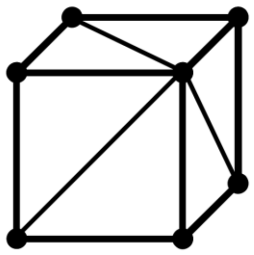
Result: The executed binary file. The source code. The report.

Theoretical part:

Let's consider a 3D object as a surface, which consists of the faces. Every face is a triangle, i.e., every face is specified by three vertices of a triangle. If we draw all faces of our object after the view and projective transformations, the result will be different from the expected. Now, our 3D model is a solid surface, and we need to separate near and far faces relatively to the observer (Fig 4.1).



a) Skeleton model



b) Solid surface model approximated by triangles

Fig 4.1

The algorithm of removing the invisible faces is required for drawing points of faces, which are visible for the observer. The algorithm of "Z-buffer" is necessary to use in this work.

Algorithm "Z-buffer"

The aim of the algorithm is constructing the depth relief of the scene. The coordinate Z contains information about the depth surface of the object. Z axis is directed to the depth of screen and matches with the observation direction.

Declare the 2D array (Z-buffer) with the size equal to the size of the screen. Fill all the cells of array with a maximal value (Z coordinates of all points of scene should be less than this value).

Calculate the Z coordinate for every point to be drawn. If coordinate is greater than value in the cell of Z-buffer (point is covered by other point) or less than 0 (point locates behind camera), then pass this point. If it is less than cell value, draw this point on the screen and update the cell value with new Z coordinate.

Calculation of value $w = 1/z$ allows us to change of algorithm conditions. All cells of the array Z-buffer should be initialized with 0. Thus, if *w < cell value* then the point is covered by other point or if $w < 0$ the point locates behind camera. The point should be drawn if w is greater than the cell value of Z buffer.

Algorithms for filling of the triangles can be used for drawing the object faces.

Calculation of the Z coordinate of a triangle point

Let's the triangle specified by three vertices A,B,C:

A=[x1,y1,z1], B=[x2,y2,z2], C=[x3,y3,z3].

Equation of surface passing through these points (determinant form):

$$\begin{vmatrix} x & y & z & 1 \\ x1 & y1 & z1 & 1 \\ x2 & y2 & z2 & 1 \\ x3 & y3 & z3 & 1 \end{vmatrix} = 0$$

where x, y, z – coordinates of the current point of the triangle.

Hence:

$$x\begin{vmatrix} y1 & z1 & 1 \\ y2 & z2 & 1 \\ y3 & z3 & 1 \end{vmatrix} - y\begin{vmatrix} x1 & z1 & 1 \\ x2 & z2 & 1 \\ x3 & z3 & 1 \end{vmatrix} + z\begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix} - \begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} = 0$$

Denote

$$\begin{vmatrix} y1 & z1 & 1 \\ y2 & z2 & 1 \\ y3 & z3 & 1 \end{vmatrix} \text{ as } a, \quad \begin{vmatrix} x1 & z1 & 1 \\ x2 & z2 & 1 \\ x3 & z3 & 1 \end{vmatrix} \text{ as } b, \quad \begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix} \text{ as } c, \quad \begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} \text{ as } d.$$

Then get the following equation:

$ax-by+cz-d=0$

Hence, the equation for *z* coordinate:

$$z = \frac{by - ax + d}{c},$$

and for *w*:

$$w = \frac{1}{z} = \frac{c}{by - ax + d}$$

Algorithm of filling of the triangle

Let’s represent the triangle as segments of straight lines (Fig 4.2).

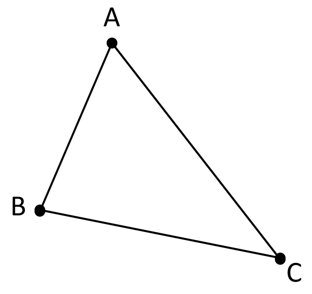


Fig 4.2

The following picture near about point A will be on the monitor screen (Fig 4.3):

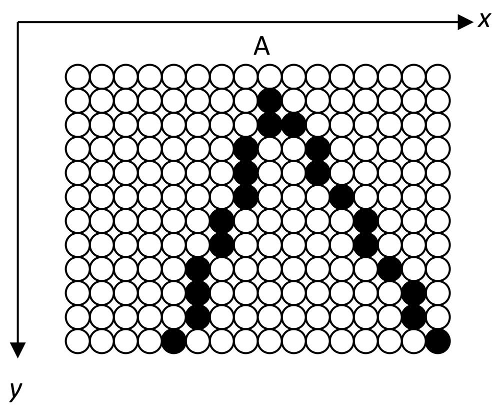
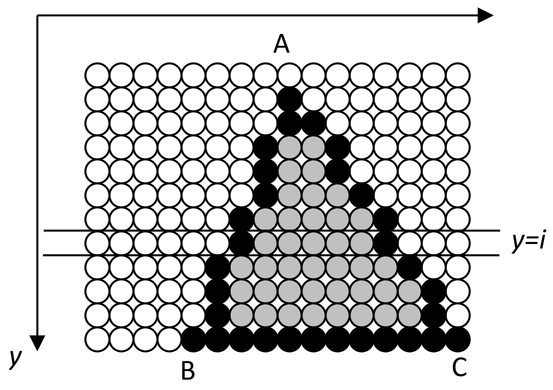


Fig 4.3

The algorithm of filling of the triangle should work as the sequence of drawing of the horizontal line segments between points of triangle with equal coordinates y.



For finding points between AB and AC you can use the Bresenham's line algorithm.

Tasks:

You need to get the parallel and perspective projections of the 3D scene with removing of invisible surfaces. The Z buffer algorithm should be used. The version of the 3D scene should be taken from previous task.