**1.1 INTRODUCTION**

3D object rotation at fixed point means first model a 3D object on 2D screen and then apply 3D rotation techniques to rotate the 3D object. This project is not limited to only rotation but there are more phases or functionalities like projection, illumination and Shading. There are different types of projection, illumination and shading, such as orthogonal projection, perspective projection and 3-point perspective projection. In this project we will use 3point perspective projection, illumination and different types of shading to give the 3D model more realistic look.

**1.2 BRIEF DESCRIPTION**

As this project is all about 3D graphics, so first of all we need to know about 3D coordinate system and 3D graphics. As we know that 2D coordinate system consists of two coordinates they are X and Y coordinates, so in 2D coordinate system or in 2D graphics, we are limited to visualize an object in two directions. That means there will be width and height of an object but there will not be depth of an object. In real world all objects have width, height and depth. In simply all the real-world objects are 3D objects and to visualize a 3D object, 2D coordinate system or 2D graphics is not applicable. So, to visualize a 3D object, we will use 3D coordinate system or 3D graphics, because in 3D coordinate system we have three coordinates namely X, Y and Z coordinates that are height, width and depth of an object. In 3D graphics we can visualize an object from any direction. In all most all the areas like entertainment, medical, games, engineering and computer-aided design industries, 3D graphics operations and techniques are the basic requirements. 3D graphics is any computer-generated visual representation of volume (in other words 3D) objects. Currently, there are various kinds of 3D graphics such as 3D rendering (it is raster computer-generated images of volume objects.), 3D animations (3D animations are animations which partially or totally consist of computer-generated volume (3D) objects. Well known are 3D animation movies produced by Hollywood and other famous production studies and 3D animation used in advertising.) and 3D interactive presentations (in other words a real-time 3D) are computer presentations which have 3D objects users can control in the real-time mode.The area of 3D graphics is a huge area and it is a rapidly increasingly area of research in scientific visualization.

Furthermore, in almost every personal computer, Mobile phones and latest technologies, 3D graphics applications and components are essential.

**1.3 AIM AND OBJECTIVE**

The main goal of this project is to develop an 3D modeling application using different 3D techniques, giving the user all freedom of viewing the object at any angle and in any orientation but provides the user with the complete model in 2D coordinate system i.e screen plane.

The project being undertaken has the following objectives: -

1. To implement the three-point perspective projection technique.
2. To implement the different 3D rotation methods.
3. To implement the 3D translation method.
4. To implement the 3D scaling method.
5. To discuss different illumination models in computer graphics.
6. To implement the flat (constant) shading technique.
7. To visualize a 3D cube on 2D screen.

**1.4 MOTIVATION**

# The 3D graphics is one of the most important components of any graphics software and 3D designing applications. There are lots of reason why 3D modeling or 3D software are important in current scenario of digital world where digital devices like computers, mobile phones and displaying devices etc. are like the essential part of any individual person. Let's take an account of the benefits because of 3D modeling or 3D graphics: -

# 3D graphics are realistic, easy, and quick. From deducing the scores of horizontal, vertical and diagonal lines of the 2D sketches to getting a one-shot picture of the architectural services is what 3D modeling facilitates.

# The design of any structure using 3D graphics and 3D games, 3D movies are better for marketing and project approvals. The experience of going through a 3D model is more compelling and satisfying to a prospect than viewing a 2D drawing. The vivid imagery lingers in the prospect's mind for a longer period of time and we stand a better chance of winning the customer. Similarly, the project approval rate in construction business is quicker when a 3D model is used.

# 3D designs are easy to re-modeling and corrections. In a 3D model it is easier to see the impact on the overall design when minor or major changes are made, this would help in finalizing the design without much cost and post-construction cost-incurring changes or corrections. It is also accurate as the end construction shapes-up to the conceived output as deduced from the 3D model.

# A 3D design can clearly show the physical dimensions of the objects and its distance in relation with other objects in the total layout. This will tremendously help customers see and adjust arrangements of objects based on their sizes to achieve varied objectives like space, movement problems, room size corrections, and so on.

# 1.5 CONCLUSION

# This chapter summarizes the first view of the undertaken project including the Introduction, Aim and Objectives, and Motivation of the project. In this chapter we have discussed the 2D and 3D coordinate system, 3D object, 3Dgraphics, 3D modelling and why and how much 3D modelling is important in current world.

**2.1 INTRODUCTION**

3D graphics and 3D modelling has been a topic of interest for research as it can be associated with different research areas like image processing and in various medical fields. During the last few decades several works has been reported where 3D modelling or 3D graphics has been widely used.

**2.2 LITERATURE REVIEW**

In 2008, Xi-Dao LUAN, Yu-Xiang XIE, Long YING and Ling-Da WU discussed about 3D modeling, IBMR, Data Acquisition, Renderingand application of 3D modelling in various fields. They also discussed about why and how much a 3D modelling or 3D graphics software system is important and some basic theoretical implementation of 3D modelling and rendering system etc. 3D modeling is a key technique in different fields of research and applications, 3D modeling methods of model-data acquisition and modeling have their own specialties. The paper systemically introduces equipments of 3D data acquisition and modeling methods, discusses the characters and developments of laser scanning system and Image-Based Modeling and Rendering (IBMR) in recent years. The paper also introduces applications of 3D modeling including tissue engineering and heritage protection. Finally, several main problems and a few deficiencies are pointed out and further challenges foreseen from three aspects: modeling retrieval, digitizing method, and dynamic modeling.

They have concluded that More and more fields now need and adopt technologies of 3D modeling. There are a number of directions in which we need to continue. Foremost among these is 3D model retrieval. To measure similarity of models, characters of 3D models, such as shape, topological construction and texture, are used. These characters are difficult to describe for users and complicated to calculate, while an effective retrieval function is necessary for an integrated 3D modeling system.

Nowadays, the visual quality becomes one of the main points of attention. There is more and more demand for 3D content with higher accuracy. Information of scene and object could not be collected absolutely during the 3D data acquisition, and some data is inevitably lost, we could not recover the real word from videos or images by the current design. So, it is worthy for us to explore new methods to digitize the real world. Dynamic model is our new direction for the future work. Dynamic models can simulate reciprocal actions of objects, which is also very helpful in exploring the discipline of thing’s evolvement.

In 2004, Mr. Suresh T. Gohane, Mr. Rushi N. Longadgedescribed the 3D Holographic projection technology and application of 3D Holographic projection technology in various fields. They also discussed about how this new technology has developed and how it come into very useful in visualization technology fields. 3D Holographic projection technology is the new sign of future technology and communications. This technology first received attention worldwide in 2008 when Prince Charles addressed the World future energy summit and made his first appearance as a hologram in a bid to reduce the royal carbon footprint. In past, American leader Al Gore launched Live Earth Tokyo in a high-tech, virtual way – as a hologram using Holographic Projection. This technology has been used widely to launch the products and create fun. The 3D holographic projection technology is also known as ''Musion Eyeliner.'' Musion Eyeliner – is a variation on the Pepper’s Ghost stage illusion. Here, the images used are three-dimensional images, but projected as two-dimensional images (2D/3D) into a 3D stage set, therefore the mind of the audience create the 3D illusion. Subjects are filmed in HDTV and broadcast on to the foil through HDTV projection systems, driven by HD Mpeg2 digital hard disc players, or uncompressed full HDTV video players. This means that production costs are minimal, needing only the single camera lenses for filming and a single projector for the playback hence the phrase ‘Glasses-free viewing’. With the different application of this technology, it will dramatically affect all the fields of life including business, education, telecommunication and healthcare.

They have concluded that Holographic Technology has endless applications as far as the human mind can imagine. Holographic Technologies are not just about art or business communication, they are about safety, security, education, planning and the strength of our civilization here and beyond. Holographic Technology will become a very integral part of human societies and civilizations in the future. This technology has recently been created to bring live holograms from one location and beam them into any location in the world.

In 2014, Er.Sheilly Padda, Er.Sonali Gupta, Er.Apoorva Arora, Er. Priya Sharmadescribed various illumination models and different types of shading methods used in 3D graphics and compared them with each other. They told that nowadays the role of images is becoming popular in the field of graphics and helpful for various researchers to carry out their research through the use of computer graphics. In this paper, they have discussed and compared the different methods used for shading an object. Shading can be applied to an object to view the different areas of the object with lighter and darker shades. They have also stated the algorithm used for phong shading and gouraud shading. From the comparison of these shading methods described here, they have concluded that phong shading is much more superior to flat and gouraud shading but requires lot of time for processing and results in better outputs.

In 2014, Filip Popovski, Igor Nedelkovski, Svetlana Mijakovskapresented a virtual environment of a real model. They also discussed about Virtual reality, Virtual environments, Scientific visualization, Computer design, Techniques of interaction and all analyses for making and visualization of virtual environment in Quest3D. All analyses of performance of the system in real time is presented. They have described advantages and disadvantages of interactions in virtual environment and made a critical analysis on a rendering speed and quality on different machines. From the performed comparison the authors conclude that, system performance is a very important feature when rendering process is in real time. If application displays min. 10 frames per second, it is considered that it is in real time. Models presented in this paper displayed 25 frames per second. In extreme situations, it can speed up to 60 frames per second. This means that the user feels comfortable in our virtual environment. From the comparison on different computer systems is conducted that working in virtual environment in real-time, computer system should have min. 1GB RAM and a solid graphics card that supports DirectX 9. Graphical cards that they have used in testing were good and with solid performance that had no problems. CPU speed should be at least 1GHz but their systems were stronger than that. Due to the high speed of rendering in some cases it has effect of motion blur.

In 2016, Georgi Golemshinskidescribed about E-learning, 3D antenna simulations in teaching physics and E-learning physics using visualizations of certain physical characteristics of the studied phenomenon are described. Teaching electromagnetism and radio-electronics, the fundamentals of radio-wave propagation and antenna construction may be realized effectively using 3D visualization, either through static or dynamic 3D graphics. The author tries to establish an e-learning route for implementing modern learning methods based on information technologies in radio physics and electromagnetism courses in universities. Authors paper discloses the benefits of implementing free access 3D software for e-learning purposes through analysing helix antenna in axial mode.

The author discussed that Understanding radio-wave propagation and antenna physics is important benefit for students in physics and radio-electronics courses. 3D visualizations of antenna characteristics help not only master antenna design and theory, but also understand the hidden insights of radio-physics and electromagnetism. The presented approach delivers to students a valuable means of performing exercises in antenna designs and thus helps the process of traditional learning of physics and radio-electronics.

In 2017, Margaret M. Fleck surveys the properties of several alternative models of image formation. A model based on stereographic projection of the viewing sphere is shown to be a better general-purpose imaging model than perspective projection. The new model can represent wider fields of view and more closely approximates real wide-angle lenses. It preserves a suitable range of shape properties, including local symmetries. It approximates narrow-angle perspective locally, enabling local affine features to be used throughout the field of view.

Perspective projection is generally accepted as the ideal model of image formation. Many recent algorithms, and many recent judgements about the relative merits of different algorithms, depend on this assumption. However, perspective projection represents only the front half of the viewing sphere and it distorts the shape and intensity of objects unless they lie near the optical axis. It is only one of several projections used in lens design and it does not accurately model the behavior of many real lenses. It works well only for narrow-angle images.

In authors paper, five alternative models for wide-angle imaging have been discussed, all of which have special properties which might be useful in particular applications. Of these, only three have low variation in intensity values and support very wide-angle images; stereographic, equi-solid angle and equidistant projection. Only stereographic and perspective projection preserve powerful shape properties useful in object recognition. Finally, because stereographic projection is conformal, it locally approximates narrow-angle perspective, so the projection of small, compact objects is approximately orthographic. This enables local affine techniques to be used anywhere in the field of view. Based on a combination of these properties, stereographic projection is the best choice for a general-purpose imaging model.

**2.3 CONCLUSION**

In this chapter various related papers have been discussed and how they have used and performed various 3D operations using their proposed algorithms and software. More emphasis is given on the 3D graphics, 3D modelling, illumination and shading.

**3.1 INTRODUCTION**

Transformations are a fundamental part of the computer graphics. Transformations are the movement of the object in cartesian plane. Transformations are used to position objects, to shape object, to change viewing positions and even how something is viewed. In simple words transformation is used for modelling and viewing. There are two types of transformation in computer graphics such as 2D transformation and 3D transformation. There are various types of 2D and 3D transformations are available in computer graphics. The some of the most important transformations are projection, translation, rotation, scaling etc. when the transformation takes place on a 3D plane, it is called as 3D transformation. 3D transformation is generalized from 2D transformation by including z coordinate.

**3.2 BRIEF DESCRIPTION OF 3D PROJECTION**

3D projection is a technique or method or 3D operation which is used to map three-dimensional coordinates to a two-dimensional coordinate system.In [3D computer graphics](https://en.wikipedia.org/wiki/3D_computer_graphics), 3D modeling is the process of developing a mathematical representation of any [surface](https://en.wikipedia.org/wiki/Surface_(mathematics)) of an objectin [three dimensions](https://en.wikipedia.org/wiki/Three-dimensional_space) via [specialized software](https://en.wikipedia.org/wiki/3D_computer_graphics_software).The product is called a 3D model. Someone who works with 3D models may be referred to as a 3D artist. It can be displayed as a two-dimensional image through a process called [3D rendering](https://en.wikipedia.org/wiki/3D_rendering) or used in a [computer simulation](https://en.wikipedia.org/wiki/Computer_simulation) of physical phenomena. The model can also be physically created using [3D printing](https://en.wikipedia.org/wiki/3D_printing) devices.3D modeling software is a class of [3D computer graphics software](https://en.wikipedia.org/wiki/3D_computer_graphics_software) used to produce 3D models. Individual programs of this class are called modeling applications or modelers.Today, 3D models are used in a wide variety of fields. The medical industry uses detailed models of organs; these may be created with multiple 2-D image slices from an [MRI](https://en.wikipedia.org/wiki/Magnetic_resonance_imaging) or [CT scan](https://en.wikipedia.org/wiki/CT_scan). The movie industry uses them as characters and objects for animated and real-life [motion pictures](https://en.wikipedia.org/wiki/Film). The [video game industry](https://en.wikipedia.org/wiki/Video_game_industry) uses them as assets for [computer and video games](https://en.wikipedia.org/wiki/Video_game). The [science](https://en.wikipedia.org/wiki/Science) sector uses them as highly detailed models of chemical compounds.

3D projection method is being used in various areas to realize or to display a 3D object in 2D coordinate system i.e screen.3D projection is especially used in engineering, computergraphics and [drafting](https://en.wikipedia.org/wiki/Technical_drawing). There are different types of projection such as orthogonal, parallel, oblique and perspective projection.

**3.2.1 PARALLEL PROJECTION**

Parallel projection is the transformation of a three-dimension area into a plane. In this projection, all projection rays are parallel. It is determined by a table (plane) and by a projection direction (vector), which cannot be parallel with the table. In accordance with the projection direction, we can split parallel projection into the following types:vertical orthogonal projectionoblique (slant). The orthogonal parallel projection into the plane xy, neglects z-coordinate. To the point P=(x,y,z) corresponding to in the projection the point P' = (x, y). The matrix representation of this transformation is:

**T =**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |

Fig. 3.2.1 (a): Transformation matrix of orthogonal projection.

The elements of this matrix are homogeneous coordinates.The matrix of homogeneous coordinates could be divided into four sub-matrixes, respectively for scaling, affine transformation, translation and projective transformation.

In [mathematics](https://en.wikipedia.org/wiki/Mathematics), homogeneouscoordinates or projective coordinates, introduced by [August Ferdinand Möbius](https://en.wikipedia.org/wiki/August_Ferdinand_M%C3%B6bius) in his 1827 work Der barycentrischeCalcülare a system of coordinates used in [projective geometry](https://en.wikipedia.org/wiki/Projective_geometry), as [Cartesian coordinates](https://en.wikipedia.org/wiki/Cartesian_coordinate_system) are used in [Euclidean geometry](https://en.wikipedia.org/wiki/Euclidean_geometry). They have the advantage that the coordinates of points, including points at infinity, can be represented using finite coordinates. Formulas involving homogeneous coordinates are often simpler and more symmetric than their Cartesian counterparts. Homogeneous coordinates have a range of applications, including [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics) and 3D [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where they allow [affine transformations](https://en.wikipedia.org/wiki/Affine_transformation) and, in general, [projective transformations](https://en.wikipedia.org/wiki/Projective_transformation) to be easily represented by a matrix.

If the homogeneous coordinates of a point are multiplied by a non-zero scalar then the resulting coordinates represent the same point. Since homogeneous coordinates are also given to [points at infinity](https://en.wikipedia.org/wiki/Point_at_infinity), the number of coordinates required to allow this extension is one more than the dimension of the [projective space](https://en.wikipedia.org/wiki/Projective_space) being considered. For example, two homogeneous coordinates are required to specify a point on the projective line and three homogeneous coordinates are required to specify a point in the projective plane.

The projects obtained in this way (meaning what we receive on the table) represents the ground plan of a projected object. In the case of projecting into the plane xz , so then neglecting the y-coordinate of the given object we obtain the elevation of an object, and in the case of projecting in the plane yz it is the side elevation of an object.

The application of homogeneous coordinates requires addition of extra n+1 coordinate for n-dimensional coordinate system. This allows not only rotation but also translation and scaling to be represented by common rotational matrix with size (n+1)x(n+1). For 3D space this leads to four-dimensional vectors usage.

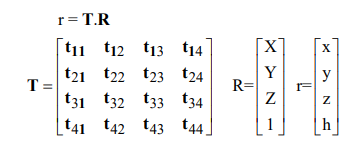


Fig. 3.2.1 (b): Homogeneous transformation matrix.

where

T is transformation matrix,

R – object space vector,

r – image space vector (mapping space).

If h is used for normalizing of r vector, then h is scale factor. In respect to this the last column and last row coefficients have the following functions:

{ti4, i = 1:3} - translation in 3D space;

{tj4,j = 1:3} projective transformation;

t44 - common scaling,

{tjj, j = 1:3} coordinate axes scaling.

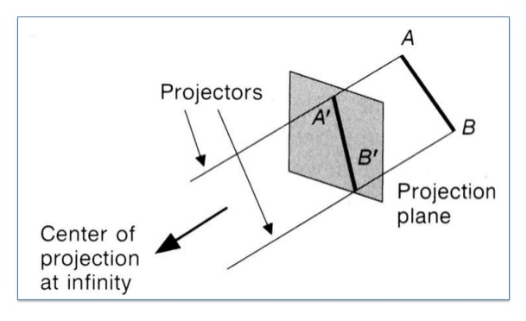


Fig. 3.2.1 (c): Example of orthogonal projection.

In the above fig. 3.2.1 (c), the line AB is projected on a projection plane using the projectors (projectors are consider as a rays) and then after the projection resultant line is A**/**B**/** as shown in the above mentioned figure. Here the center of projection is infinity.

* + 1. **PERSPECTIVE PROJECTION**

Perspective projection is given by a table and a center of projection. All projection rays start from one point called the center of projection. This type of projection creates an image similar to that seen by the human eye in the real world. An object standing farther away appears smaller. This type of projection does not maintain parallelism of abscissas. Projects of the parallel abscissas in the 3D area are skew abscissas, except the case where the abscissa lies in the plane parallel to the table.

Among all the projection types, perspective projection gives the best 3D visualization of a 3D object.In perspective projection, the distance between the project plane and center of projection is finite and the object size varies inversely with respect to distance which gives more realistic look.The plane in the area into which we transform (project) objects is called the 'Table'.

So, in this application perspective projection had been used. Again, perspective projection has its own three categories such as one-point perspective projection, two-point perspective projection and three-point perspective projection. Among the types of perspective projection three-point perspective projection is the most complex method to visualize an object but it gives more realistic view of an object. In all the types of perspective projection, the number one, two and three, defines the number of vanishing points. Certain set of parallel lines appear to meet at a different point, called the vanishing point for this direction. Principal vanishing points are formed by the apparent intersection of lines parallel to one of the three principal x,y,z axes. The number of principal vanishing points is determined by the number of principal axes intersected by the view plane. Sets of parallel lines on the same plane lead to collinear vanishing points. The line is called the horizon for that plane.Vanishing point is a point on the view plane where lines appears to converge at that point. In simple, we can consider a vanishing point is a point or position from where a viewer is viewing the object. So, in three-point perspective projection, there are three vanishing points.

The transformation matrix or formula of three-point perspective projection can be represented in the matrix form as shown in below figure. The elements of the three-point perspective projection are given as homogeneous coordinates in 4 x 4 matrix.

**T =**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 1/dx |
| 0 | 1 | 0 | 1/dy |
| 0 | 0 | 1 | 1/dz |
| 0 | 0 | 0 | 1 |

**T =**

Fig. 3.2.2 (a):transformation matrix of 3-point perspective projection

In 3D projection, 3D coordinates are transformed into 2D coordinates using perspective projection method. In the above figure 3.2.2(a) the 3point perspective matrix is showing in which instead of using three coordinates, four coordinates are taken, which is known as homogeneous coordinates. 1/dx, 1/dy, and 1/dz are the vanishing points in 3point perspective projection.

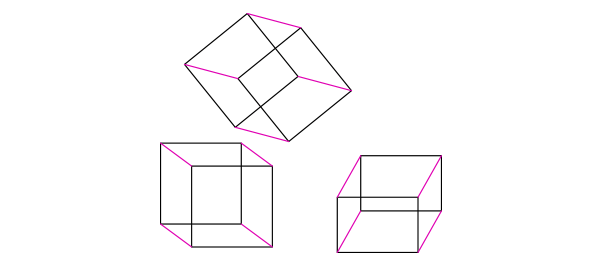


Fig. 3.2.2 (b): perspective view of a cube

In the above figure 3.2.2(b), different perspective views of a cube have been shown.

**3.2.2.1ONE- POINT PERSPECTIVE PROJECTION**

A drawing has one-point perspective when it contains only one vanishing point on the horizon line. This type of perspective is typically used for images of roads, railway tracks, hallways, or buildings viewed so that the front is directly facing the viewer. Any objects that are made up of lines either directly parallel with the viewer's line of sight or directly perpendicular (the railroad slats) can be represented with one-point perspective. These parallel lines converge at the vanishing point.

One-point perspective exists when the [picture plane](https://en.wikipedia.org/wiki/Picture_plane) is parallel to two axes of a rectilinear (or Cartesian) scene – a scene which is composed entirely of linear elements that intersect only at right angles. If one axis is parallel with the picture plane, then all elements are either parallel to the picture plane (either horizontally or vertically) or perpendicular to it. All elements that are parallel to the picture plane are drawn as parallel lines. All elements that are perpendicular to the picture plane converge at a single point (a vanishing point) on the horizon. The formula of one-point perspective projection can be represented in matrix form as below-

**T =**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1/d |
| 0 | 0 | 0 | 1 |

Fig. 3.2.2.1 (a):Transformation matrix of one-point perspective projection

In 3D projection, 3D coordinates are transformed into 2D coordinates using perspective projection method. In the above figure 3.2.2.1(a) the 3point perspective matrix is showing in which instead of using three coordinates, four coordinates are taken, which is known as homogeneous coordinates. Where, d is the direction or point of projection or vanishing point. This vanishing point can be on any of the three X, Y and Z plane.

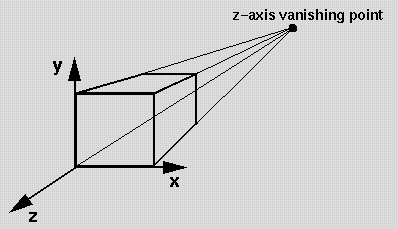


Fig. 3.2.2.1(b):One- point perspective projection

In the above figure 3.2.2.1(b) one-point perspective projection of a cube has been showing in which there is only one vanishing point on z plane. Vanishing point is the point where the lines are converged together.

**3.2.2.2TWO-POINT PERSPECTIVE PROJECTION**

A drawing has two-point perspective when it contains two [vanishing points](https://en.wikipedia.org/wiki/Vanishing_point) on the horizon line. In an illustration, these vanishing points can be placed arbitrarily along the horizon. Two-point perspective can be used to draw the same objects as one-point perspective, rotated: looking at the corner of a house, or at two forked roads shrinking into the distance, for example. One point represents one set of [parallel lines](https://en.wikipedia.org/wiki/Parallel_lines), the other point represents the other. Seen from the corner, one wall of a house would recede towards one vanishing point while the other wall recedes towards the opposite vanishing point.Two-point perspective exists when the painting plate is parallel to a [Cartesian scene](https://en.wikipedia.org/w/index.php?title=Cartesian_scene&action=edit&redlink=1) in one axis (usually the [z-axis](https://en.wikipedia.org/wiki/Z-axis)) but not to the other two axes. If the scene being viewed consists solely of a cylinder sitting on a horizontal plane, no difference exists in the image of the cylinder between a one-point and two-point perspective.

Two-point perspective has one set of lines parallel to the picture plane and two sets oblique to it. Parallel lines oblique to the picture plane converge to a vanishing point, which means that this set-up will require two vanishing points.The formula of two-point perspective projection can be represented in matrix form as below-

**T =**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 1/dx |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1/dy |
| 0 | 0 | 0 | 1 |

Fig.3.2.2.2 (a): Transformation matrix of two-point perspective projection

In two-point perspective projection, 3D coordinates are transformed into 2D coordinates using perspective projection method. In the above figure 3.2.2.2 (a) the two-point perspective matrix is showing in which instead of using three coordinates, four coordinates are taken, which is known as homogeneous coordinates. 1/dx and 1/dy are the vanishing points in two-point perspective projection.Where, dx is the vanishing point on x-axis direction and dy is the vanishing point on y-axis direction.

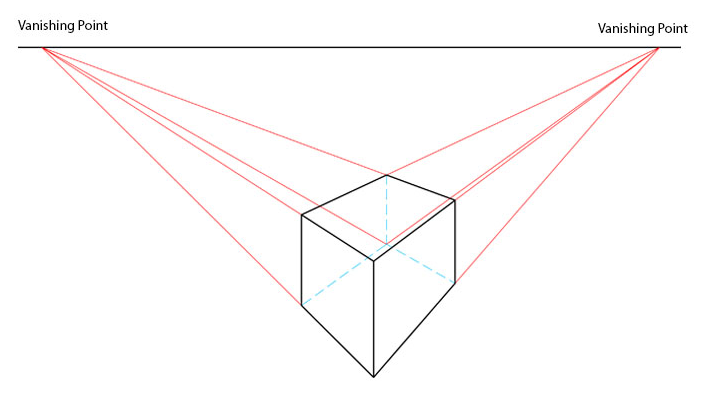


Fig. 3.2.2.2 (b): Two-point perspective projection

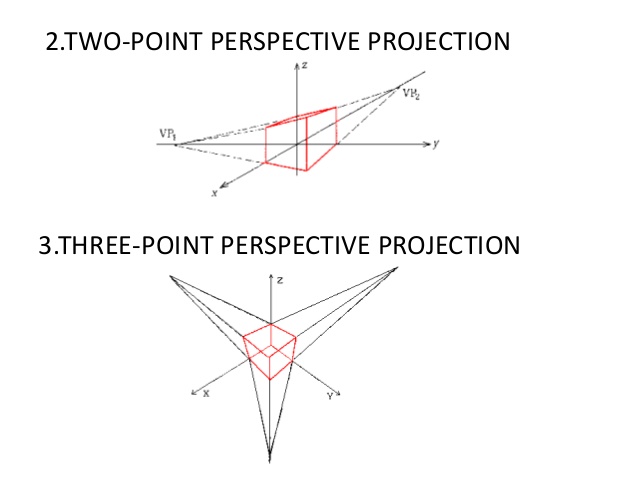
In the above figure 3.2.2.2(b) two-point perspective projection of a cube has been showing in which there are two vanishing points on a horizon line. Vanishing point is the point where the lines are converged together.

**3.2.2.3THREE -POINT PERSPECTIVE PROJECTION**

Three-point perspective is often used for cuboidal shape like buildings seen from above (or below). In addition to the two vanishing points from before, one for each wall, there is now one for how the vertical lines of the walls recede. For an object seen from above, this third vanishing point is below the ground. For an object seen from below, as when the viewer looks up at a tall building, the third vanishing point is high in space.

Three-point perspective exists when the perspective is a view of a Cartesian scene where the picture plane is not parallel to any of the scene's three axes. Each of the three vanishing points corresponds with one of the three axes of the scene. One, two and three-point perspectives appear to embody different forms of calculated perspective, and are generated by different methods. Mathematically, however, all three are identical; the difference is merely in the relative orientation of the rectilinear scene to the viewer. Characteristic of three-point perspective projection are as follows:

1. The center of projection (CP) is a finite distance from an object.
2. In this projection, projectors are rays (i.e., non-parallel).
3. It has three vanishing points.
4. It is a type of projection in which objects appear smaller as distance from center of projection (say eye of an observer) increases and objects appear bigger as distance from center of projection decreases.
5. It does not maintain the exact size and shape of an object, but it gives the realistic visualization of an object. That is why in this projection determining the size and shape of an object is difficult.
6. It is the most realistic representation of an object and difficult to execute.



VP1

VP2

VP3

Fig. 3.2.2.3: Three-point perspective projection

In the above figure 3.2.2.3 three-point perspective projection of a cube has been showing in which there are three vanishing points denoted by VP1, VP2 and VP3. Vanishing point is the point where the lines are converged together.

* 1. **CONCLUSION**

# This chapter summarizes the different 3D projection methods. In this chaptermore emphasis has been given onone-point, two-point and three-point perspective methods and their related formulae and diagrams.

**4.1 INTRODUCTION**

3D rotation is a 3D transformation of a 3D object. It changes the orientation of a 3D object.3D rotation is different from 2D rotation, as in 2D rotation, objects can be rotated in two directions only but 3D rotation, objects can be rotated in three directions i.e can be rotated in all directions. To perform 3D rotation, we have to specify or enter the angle of rotation along with the axis of rotation. 3D rotation canbe performed about X, Y, and Z axes.3D Rotation is more complicated than 2D rotation since we must specify an axis of rotation. In 2D the axis of rotation is always perpendicular to the xy plane, i.e., the Z axis, but in 3D the axis of rotation can have any spatial orientation. We will first look at rotation around the three principle axes (X, Y, Z) and then about an arbitrary axis.

The formula of X, Y, and Z axes rotation can be represented in the matrix form as below: -

|  |  |  |  |
| --- | --- | --- | --- |
| cos**θ** | 0 | sin**θ** | 0 |
| 0 | 1 | 0 | 0 |
| -sin**θ** | 0 | **cosθ** | 0 |
| 0 | 0 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | cos**θ** | -sin**θ** | 0 |
| 0 | sin**θ** | **cosθ** | 0 |
| 0 | 0 | 0 | 1 |

**Rx(θ)=**

**Ry (θ)=**

**Rx (θ)=**

fig. 4.1(b): transformation matrix of y-axis rotation

fig. 4.1(a): transformation matrix of x-axis rotation

|  |  |  |  |
| --- | --- | --- | --- |
| cos**θ** | -sin**θ** | 0 | 0 |
| sin**θ** | cos**θ** | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |

**Rz(θ)=**

fig. 4.1(c): transformation matrix of z-axis rotation

Rotation about any axis will be in two directions that are clockwise and anticlockwise. When the sign of the angle is positive, then rotation will be in anticlockwise direction and if the sign of the angle is negative, then rotation will be in clockwise direction. To perform the rotation about all direction at a time, we have to combine these rotation formulae together so as to get an integrated formula. If someone wants to rotate an object in three direction, then the object will first be rotated in z-direction, then in y-direction and at last rotation will be in x-direction. So, it is essential to put all the rotation formulae properly in sequence from right first z-axis rotation formula followed by y-axis rotation and the x-axis rotation formula.

In 3D rotation operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), rotation angles are θ for all respectively, and the produced coordinates are (X’, Y’, Z’). This can be mathematically represented as shown below –

**4.2** [**Z-AXIS ROTATION**](https://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/3drota.htm#Z)

Z-axis rotation is identical to the2D case about origin for which we have derived the matrices already. In z-axis rotation, the x and y coordinate will be transformed but the z-coordinate will remain same asdiscussed below-

|  |  |  |  |
| --- | --- | --- | --- |
| cos**θ** | -sin**θ** | 0 | 0 |
| sin**θ** | cos**θ** | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |

**Rz(θ)=**

|  |  |  |
| --- | --- | --- |
| x' | y' | z' |

P' =

|  |  |  |
| --- | --- | --- |
| x | y | z |

P =

* P' = P \* Rz(**θ)** 🡪(1)

x' = x\*cos **θ** - y\*sin **θ** 🡪(2)  
y' = x\*sin **θ** + y\*cos**θ** 🡪(3)  
z' = z 🡪(4)

The equation (1) is the formula for z-axis rotation, equation (2) is the transformation equation for x coordinate, equation (3) is the transformation equation for y coordinate and equation (4) is the transformation equation for z coordinate which will remain same in z-axis rotation.

Where,

P is the original coordinate.

Rz(θ)is the transformation rotation matrix for z-axis rotation.

P'is the transformed coordinate.

**θ** is the angle of rotation about z-axis.

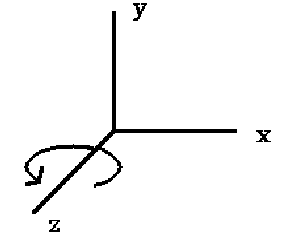


Fig. 4.2: Z-axis rotation

In the above figure 4.2, line segments represent the x, y and z axes. The curved arrow represents the rotation direction about z-axis.

**4.3** [**X-AXIS ROTATION**](https://www.siggraph.org/education/materials/HyperGraph/modeling/mod_tran/3drota.htm#X-Axis%20Rotation)

X-axis rotation looks like Z-axis rotationabout origin if replace:

X axis with Y axis  
Y axis with Z axis  
Z axis with X axis

In x-axis rotation, the z and y coordinates will be transformed but the x-coordinate will remain same. For which we have derived the matrices as follows: -

In 3D rotation operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), rotation angles are θ for all respectively, and the produced coordinates are (X’, Y’, Z’). This can be mathematically represented as shown below –

|  |  |  |
| --- | --- | --- |
| x | y | z |

|  |  |  |
| --- | --- | --- |
| x' | y' | z' |

P' = P =

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | cos**θ** | -sin**θ** | 0 |
| 0 | sin**θ** | **cosθ** | 0 |
| 0 | 0 | 0 | 1 |

**Rx(θ)=**

* P' = P \* Rx(θ) 🡪(1)

y' = y\*cos q - z\*sin q 🡪(2)   
z' = y\*sin q + z\*cos q 🡪(3)

x' = x 🡪(4)

The equation (1) is the formula for x-axis rotation, equation(2) is the transformation equation for y coordinate, equation (3) is the transformation equation for z coordinate and equation (4) is the transformation equation for x coordinate which will remain same in x-axis rotation.

Where,

P is the original coordinate.

Rx(θ)is the transformation rotation matrix for x-axis rotation.

P'is the transformed coordinate.

**θ** is the angle of rotation about x-axis.

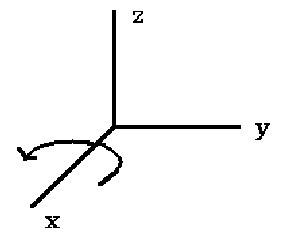


Fig. 4.3: X-axis rotation

In the above figure 4.3, line segments represent the x, y and z axes. The curved arrow represents the rotation direction about x-axis.

**4.4 Y-AXIS ROTATION**

Y-axis rotation looks like Z-axis rotation if we replace the following:

X axis with Z axis  
Y axis with X axis  
Zaxis with Y axis

In y-axis rotation, the x and z coordinates will be transformed but the y-coordinate will remain same. For which we have derived the matrices as follows: -

In 3D rotation operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), rotation angles are θ for all respectively, and the produced coordinates are (X’, Y’, Z’). This can be mathematically represented as shown below –

|  |  |  |  |
| --- | --- | --- | --- |
| cos**θ** | 0 | sin**θ** | 0 |
| 0 | 1 | 0 | 0 |
| -sin**θ** | 0 | **cosθ** | 0 |
| 0 | 0 | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| x | y | z |

P =

**Ry(θ)=**

|  |  |  |
| --- | --- | --- |
| x | y | z |

P' =

* P' = P \* Ry(θ) 🡪(1)

z’ = z\*cos q – x\*sin q 🡪(2)   
x’ = z\*sin q + x\*cos q 🡪(3)   
y’ = y 🡪(4)

The equation (1) is the formula for y-axis rotation, equation (2) is the transformation equation for z coordinate, equation (3) is the transformation equation for x coordinate and equation (4) is the transformation equation for y coordinate which will remain same in y-axis rotation.

Where,

P is the original coordinate.

Rx(θ)is the transformation rotation matrix for x-axis rotation.

P'is the transformed coordinate.

**θ** is the angle of rotation about x-axis.

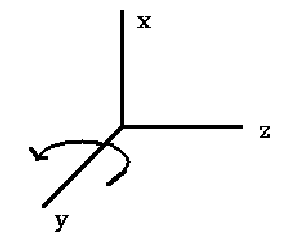


Fig. 4.4: Y-axis rotation

In the above figure 4.4, line segments represent the x, y and z axes. The curved arrow represents the rotation direction about y-axis.

## **4.5 ROTATION ABOUT AN ARBITRARY AXIS**

This is similar to 2D rotation about an arbitrary point. To perform the rotation about an arbitrary axis we first need to perform transformation which align rotation axis with one of coordinate axis (x, y, z) and then perform rotation about the axis, finally perform the inverse transformation of as first operation. When an object is rotated in 2D plane, the object moves from its original position, so to perform rotation at fixed point, we first perform negative translation to take the object to the origin and then perform rotation about any axis, after that perform positive translation to bring the object to its original position.

To perform rotation about an arbitrary axis, we need to integrate all the rotation matrices together as follows: -

|  |  |  |
| --- | --- | --- |
| x | y | z |

|  |  |  |
| --- | --- | --- |
| x | y | z |

P = P' =

|  |  |  |
| --- | --- | --- |
| tx | ty | tz |

|  |  |  |
| --- | --- | --- |
| -tx | -ty | -tz |

T = T' =

* P**'** = P \* T \* Rx (θ) \* Ry (θ) \* Rz(θ) \*T**'** 🡪(1)
* X' = x (cy\*cz) + y (-cy\*sz) + z (sy) + cy (-centerx\*cz + centery\*sz) – sy (centerz) +

Centerx 🡪(2)

* Y' = x (sx\*sy\*cz ) + y (-sx\*sy\*sz + cx\*cz) + z (-sx\*cy) + cx (-centerx\*sz - centery\*cz) –

sx (-sy\*(-centerx\*cz + centery\*sz) – cy\*centerz) + centery))🡪(3)

* Z' = x (sx\*sz-sy\*cx\*cz ) + y (sx\*cz + cx\*sy\*sz) + z (cx\*cy) + sx (-centerx\*sz–

centery\*cz) – cx (-sy \*(-centerx\*cz + centery\*sz) – cy\*centerz) + centerz))🡪(4)

The equation (1) is the formula for integrated rotation about all axes, equation (2) is the transformation equation for x coordinate, equation (3) is the transformation equation for y coordinate and equation (4) is the transformation equation for z coordinate in rotation about all axes.

Where,

P is the original coordinate.

P' is the transformed coordinate.

T is the positive translation matrix.

T' is the negative translation matrix.

X, Y, Z are the original coordinates.

Tx, ty, tz are the translation factors.

X', Y', Z' are the transformed coordinates after rotation.

Rx (θ) is rotation matrix of x-axis rotation

Ry (θ) is rotation matrix of y-axis rotation

Rz(θ) is rotation matrix of z-axis rotation

Sx is the sine of angle of x-axis rotation.

Cx is the cosine of angle of x-axis rotation.

Sy is the sine of angle of y-axis rotation.

Cy is the cosine of angle of y-axis rotation.

Sz is the sine of angle of z-axis rotation.

Cz is the cosine of angle of z-axis rotation.

Centerx, centery and centerz are the coordinates of center of object.

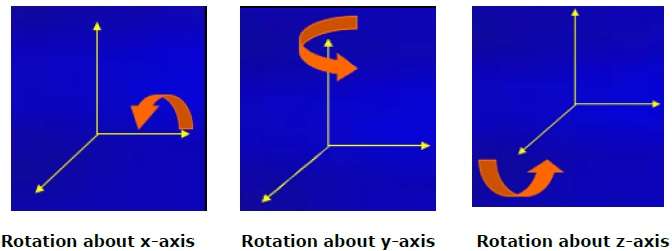


Fig. 4.5: Rotation in x, y and z-axis

The figure 4.5 show the rotation about x-axis, y-axis and z-axis. The line segments represent the x, y and z axes. The curved arrow represents the direction of rotation about x-axis, y-axis and z-axis.

**4.6 CONCLUSION**

# This chapter summarizes the different 3D rotation methods. In this chapter more, emphasis has been given on formulas of x-axis, y-axis and z-axis rotation and their related diagrams.

**5.1 INTRODUCTION**

In computer graphics, translation and scaling are the two basic transformations which are used much often in 3D modelling works. Translation is the transformation operation in which an object is moved from position to another position in 2D plane. Whereas scaling is the transformation operation that changes the size of an object and repositions the object relative to the origin coordinate.

**5.2 3D TRANSLATION**

In 3D translation, we transfer the Z coordinate along with the X and Y coordinates. The process for translation in 3D is similar to 2D translation. A translation operation moves an object into a different position on the screen.

The following figure shows the effect of translation-

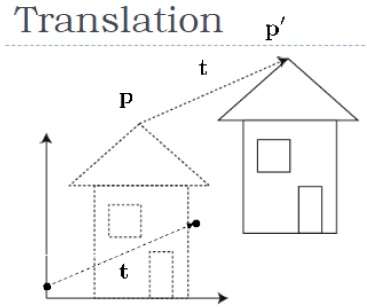


Fig. 5.2(a): 3D translation

A point can be translated in 3D by adding translation coordinate (tx,ty,tz)to the original coordinate P(X, Y, Z) to get the new coordinate P՜(X’, Y’, Z’). 3D translation transformation matrix can be represented as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| tx | ty | tz | 1 |

**T =**

Fig. 5.2(b): 3D translation matrix

The 3D translation matrix is a 4x4 because, in 3D translation homogeneous coordinates are used. Here tx, ty and tz are the translation factors.

**5.3 3D SCALING**

Since, in this project we have taken a cube as 3D object. So, a cube is made up of six faces and each face is represented on computer screen using pixels, if the size of the faces is small, then it is very difficult to visualize so computer screen. So, to enlarge the size of the cube, we have implemented the concept of scaling. Scaling is the 3D transformation technique, which can be used to increase or decrease the size of an object to visualize the object more clearly on a computer screen. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. The following figure shows the effect of 3D scaling.

In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), scaling factors are (Sx,Sy,Sz)respectively, and the produced coordinates are (X’, Y’, Z’). 3D scaling transformation matrix can be represented as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Sx | 0 | 0 | 0 |
| 0 | Sy | 0 | 0 |
| 0 | 0 | Sz | 0 |
|  |  |  |  |
| 0 | 0 | 0 | 1 |

**S =**

Fig. 5.3(a): transformation matrix of 3D scaling

The 3D scaling matrix is a 4x4 matrix because, in 3D scaling homogeneous coordinates are used. Here sx, sy and sz are the scaling factors. Scaling factors can be any integers positive or negative. If the scaling factor is positive then the object will be enlarged and if the scaling factor is negative then the object will be compressed.

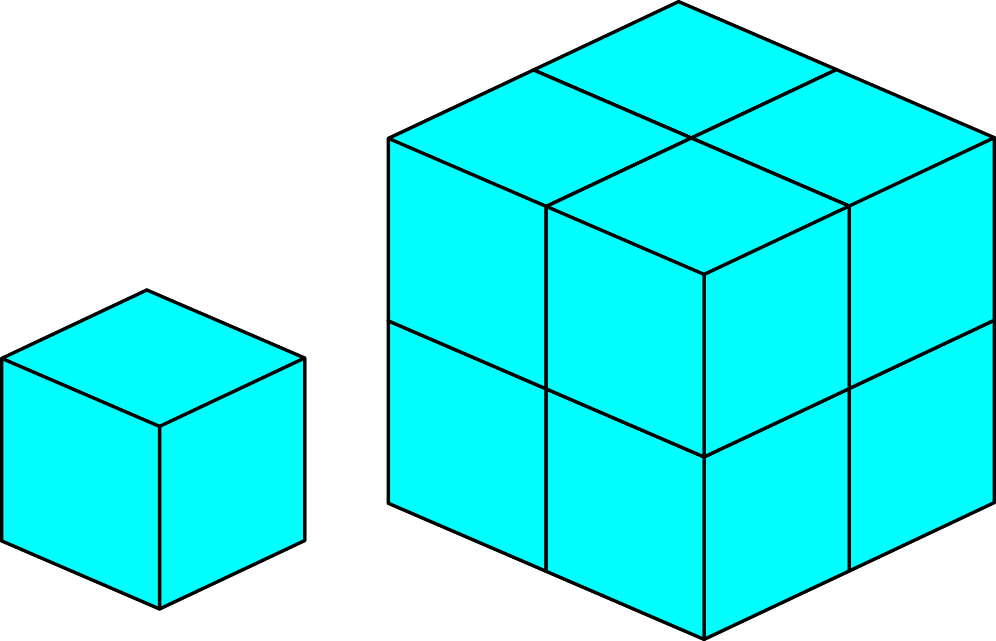
****

Fig. 5.3(b):3D scaling of a cube

In the above figure 5.3(b), scaling of a cube is showing in which left-hand sides smaller cube is the original cube and the right-hand sides bigger cube is the transformed cube i.e enlarged cube from the small cube. In this scaling operation the scaling factor is of 4, that is the cube is enlarged four times than the original one.

**5.4 CONCLUSION**

This chapter summarizes the different 3D translation and scaling methods. In this chapter more, emphasis has been given on formulas of translation and scaling operations and on their related diagrams.

**6.1 INTRODUCTION**

Illumination (lighting) model: determine the color of a surface point by simulating some light attributes. To model the interaction of light with surfaces to determine the final color &brightness of the surface. There are two types of illumination to determine the effect of light they are as follows -

1. Global illumination
2. Local illumination

Global Illumination models: take into account the interaction of light from all the surfaces in the scene. On the other hand, local illumination only considers the light, the observer position, and the object material properties.

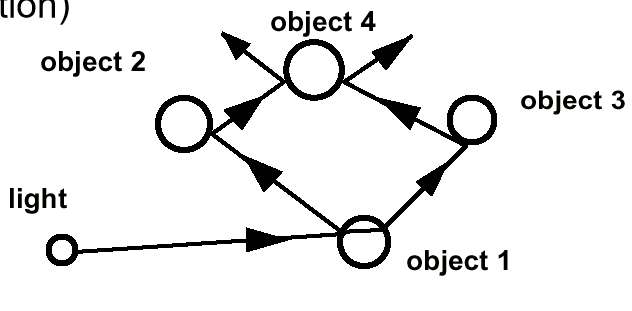
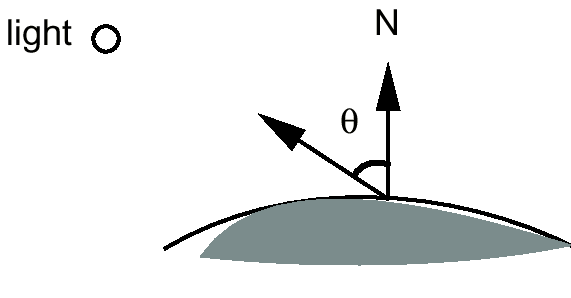


Fig. 6.1(a):Global illumination

The above figure 6.1(a), shows the reflection of lights from multiple objects. In the figure, light is the position of a light source which emits light in all direction. Light rays are fall on the object 1 and then reflected to the object 2 and object 3, again light reflected from object 2 and object 2 to object 4. So in global illumination we have to consider all reflected lights to determine the effect of light on a surface of an object.

****

**R**

Fig. 6.1(b):Local illumination

The above figure 6.1(b) shows the reflection of light from an object. In the figure, light O is the position of a light source which emits light in all direction. Light rays are fall on a surface of an object and then reflected. In this figure N is the normal of the surface and R is the reflected ray and theta is the angle between the normal and reflected ray vectors. So, in local illumination we have to determine the normal vector and reflected ray vector of the surface to determine the effect of light on the surface of an object.

Simple and fast method for calculating surface intensity at a given point. Lighting calculations are based on the background lighting conditions, the light source specification, color, position, Optical properties of surfaces: Glossy OR matte, Opaque OR transparent (control refection and absorption). There are three basic illumination models as follows –

**6.2** **AMBIENT LIGHT (BACKGROUND LIGHT)**

The light that is the result from the light reflecting off other surfaces in the environment. A general level of brightness for a scene that is independent of the light positions or surface directions. Ambient light has no direction. Each light source has an ambient light contribution, Ia. For a given surface, we can specify how much ambient light the surface can reflect using an ambient reflection coefficient: Ka (0 < Ka< 1). So, the amount of light that the surface reflects is therefore:

Iamb = KaIa

Where,

Ka is the ambient reflection coefficient which in between 0 and 1.

Ia is the ambient light contribution by a light source.

Iambambient light reflection from the surface of an object.

**6.3 DIFFUSE LIGHT**

The illumination that a surface receives from a light source and reflects equally in all directions. Diffuse reflection (or Lambertian Reflection) is exhibited by dull, matte surfaces (e.g. chalk, unfinished wood, carpet). The brightness depends only on the lightdirection(L) and surface normal(N). So, the amount of light that the surface reflects is therefore:

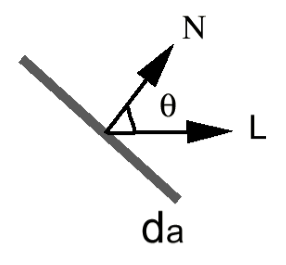


Fig. 6.3(a): Direction of light in diffuse reflection

The above figure 6.3(a) shows the reflection of diffuse light from an object. Light rays are fall on a surface of an object and then reflected in the direction of L. In this figure N is the normal of the surface,L is the reflected ray, da is the diameter of diffuse light and theta is the angle between the normal and reflected ray vectors. So, in diffuse reflection we have to determine the normal vector and reflected ray vector of the surface to determine the effect of light on the surface of an object.

Idiff = Kd Id cos(θ)

If N and L are normalized, cos(θ) = N . L

Idiff = Kd Id (N **.** L)

Where,

Kd is the diffuse reflection coefficient which in between 0 and 1.

Id is the diffuse light contribution by a light source.

Idiff is the diffuse light reflection from the surface of an object.

N is the normal vector of a plane

L is the light source vector

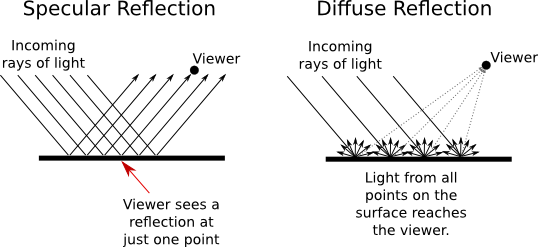


Fig. 6.3(b):Example of diffuse reflection

The above figure 6.3(b) shows the example of diffuse reflection where bold line segments are the incoming light rays of light, dotted lines are the diffuse reflection rays of light and the big dot represents the position of viewer who is seeing the object.

**6.4 SPECULAR LIGHT**

These are the bright spots (specular highlights) on objects (such as polished metal, apple)Light reflected from the surface unequally to all directions.How much reflection light we can see depends on where we are? So, the amount of light that the surface reflects is therefore:

Ispec = Ks Iscosn(θ)

Where,

Ks is the specular reflection coefficient

Is is specular component of the light source.

Ispecis the specularlight reflection from the surface of an object.

N is the normal vector of a plane

L is the light source vector

V is the reflection light vector seen by the viewer.

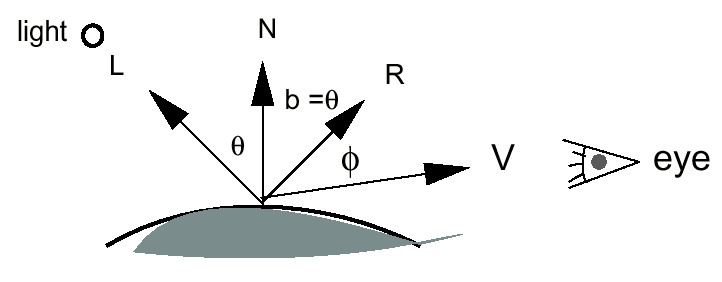


Fig. 6.4(a) Direction of light in specular reflection

The above figure 6.4(a) shows the reflection of specular light from an object. Light rays are fall on a surface of an object and then reflected in the direction of L, R and V. In this figure N is the normal of the surface, L, R and V are the reflected rays, theta and phi are the angle between the normal and reflected ray vectors L and reflected ray R and reflected ray V respectively. In specular reflection the viewer can see the reflected ray V as shown in the above figure. So, in specular reflection we have to determine the normal vector N and reflected ray vector V of the surface to determine the effect of light on the surface of an object.

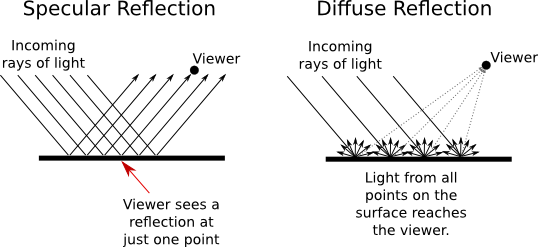


Fig. 6.4(b):Example of light in specular reflection

The above figure 6.4(b) shows the example of specular reflection where line segments without arrow heads are the incoming light rays of light and lines with arrow heads are the specular reflection rays of light. In specular reflection viewer sees a reflection at just one point.

When we combined these models, we get a single light (white light source).

I = Iamb + Idiff + Ispec

* I = KaIa +Kd IL (N . L) + Ks IL (R . V)n

The combined model is known as Phong model.

Fig. 6.4 (c):All type of lights

**6.5 CONCLUSION**

# This chapter summarizes the different illumination models. In this chapter more emphasis has been given on ambient light, diffuse light and specular lights models and their related formulae and diagrams. In this chapter, what are the condition and what are basic calculations needs to perform to determine the effect of various lighting model has been discussed.

**7.1 INTRODUCTION**

Shading is theapplication of illumination models at a set of points and colors the whole image. There are different types of shading models for polygons such as Gouraud shading, Phong shading, Flat (constant shading). Flat shading is also known as constant shading. In flat shading, illumination is calculated at any one point on the surface and apply on the whole surface. To calculate illumination, we can take face or one normal from a pair of edges. Good for far away light and viewer or if facets approximate surface well. In interpolated shading, illumination is need to calculate at vertices and interpolate color and in per-pixel shading, illumination is calculated at every point on the surface. But in this application, we have implemented the concept of flat shading.

**7.2 GOURAUD SHADING**

Gouraud shading, named after Henri Gouraud, also called as 'intensity interpolation shading' or 'color interpolation shading' is a method of interpolation used for shading surfaces represented by different polygons. Gourard shading explains the concept of interpolation which is applied to individual polygon vertices. The Gouraud shading process requires that the normal be known for each vertex of the polygonal mesh. This algorithm computes the intensities at each vertex of polygonal mesh and then interpolates the intensities across the polygon. The gouraud shading computes the intensity calculations down and then across each scan lines thus eliminating the sharp edges. The main purpose of Gouraud shading is to eliminate the discontinuities in intensity along polygon edges.

The algorithm for gouraud shading is as follows:

1. Calculate intensities at each vertex of polygon (Nv). To calculate the intensity at each vertex, we need to calculate unit normal vector at the vertex
2. Interpolates these vertex intensities along the edges of polygon.
3. Interpolate these edge intensities along the scan line in the interior of the polygon.

**7.3 FLAT SHADING**

Flat shading is the fastest and simple method of shading but does not god quality of images. Flat shading (constant shading) requires calculation of intensities for each polygon surfaces. Flat surface rendering or constant shading is the simplest rendering format that involves some basic surface properties such as color distinctions and reflectivity. This method produces a rendering that does not smooth over the faces which make up the surface. The resulting visualization shows an object that appears to have surfaces faceted like a diamond. The disadvantage of flat shading is that it gives low-polygon models a faceted look. Sometimes this look can be advantageous though, such as in modeling boxy objects. Artists sometimes use flat shading to look at the polygons of a solid model they are creating. Rendering only requires the computation of a color for each visible face. The whole face is filled with this color. This approach is fast and very simple, but it gives quite unrealistic results and non-smooth surfaces. This is highlighted by the Mach effect: the intensity at the vicinities of the edges is overestimated for light values and underestimated for dark values. Below figure shows an object shaded using flat shading.

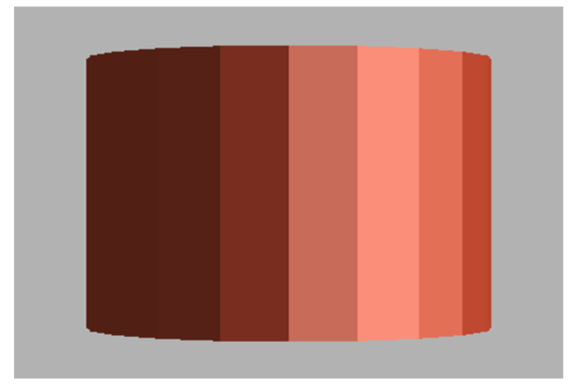


Fig. 7.3(a): Example of flat shading

In flat shading each polygon is drawn with the same color. We just need to know one normal for the entire surface. Given a single normal to the plane the lighting equations and the material properties are used to generate a single color. The polygon is filled with that colour.But in gouraud shading we need to know the normal at each vertex of the polygon and this method is slower than flat shading. Fig.4 shows the comparison of flat and gouraud shading.

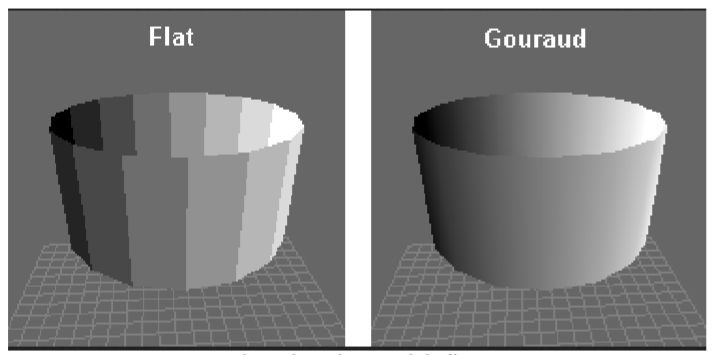


Fig. 7.3 (b): Flat and Gouraud shading

The figure 7.3(b) shows the difference between flat shading and gouraud shading. As we can see in picture that in flat shading, it is clearly visible that the cylinder is divided into some rectangular shapes and then shading has been performed and it is lead to rough or blur appearance of the object. But in the gouraud shading it is not visible that the object is divided into some rectangular shapes and then shading is performed and in gouraud shading the appearance of the object is smoother than the flat shaded object.

**7.4 PHONG SHADING**

Phong shading is one of the most useful shading algorithms in computer generated images as it provides high degree of practicality. It is slowest but provides best quality of images. The first stage in the process is the same as for the Gouraud Shading - for any polygon we evaluate the vertex normals. For each scan line in the polygon we evaluate by linear interpolation the normal vectors at the end of each line. These two vectors Na and Nb are then used to interpolate Ns. We thus derive a normal vector for each point or pixel on the polygon that is an approximation to the real normal on the curved surface approximated by the polygon. Ns, the interpolated normal vector, is then used in the intensity calculation. The vector interpolation tends to restore the curvature of the original surface that has been approximated by a polygon mesh. Phong shading species how to calculate color on every point of surface. But calculating of color at every point may be slow therefore phong shading employs calculating intensities at vertices only and then apply interpolation to calculate in between points (between pixels).

**The algorithm is as follows:**

1. Compute a normal N for each vertex of the polygon.
2. From bi-linear interpolation compute a normal, Ni for each pixel.
3. From Ni compute intensity Ii for each pixel of the polygon.
4. Paint pixel to shade corresponding to Ii.

**7.5 COMPARISON OF SHADING METHODS**

|  |  |  |
| --- | --- | --- |
| **Flat shading** | **Gouraud shading** | **Phong shading** |
| Also called constant shading. Computes illumination once per polygon and apply it to whole polygon. | Computes illumination at border (vertices) and interpolates. | Also called accurate shading. Applies illumination at every point of polygon surface. |
| Creates discontinuities in colour. | Interpolates colours along edges and scanline. | Interpolates normal instead of colours. |
| Problem of mach-bands. | Handles mach-bands problem found in flat shading. | Removes Mach-bands completely. |
| Low cost. | Not so expensive. | More expensive than gouraud shading. |
| Requires very less processing and is fast in time. | Requires moderate processing and time. | Requires complex processing and is slower but is more efficient as compared to other shading methods. |
| Lighting equation used once for polygon. | Lighting equation used at each vertex. | Lighting equation used at each pixel. |

Fig. 7.5: Comparison table of shading methods

**7.6 CONCLUSION**

This chapter summarizes the different shading methods. In this chapter more emphasis has been given on flat shading, gouraud shading and phong shading, also their related formulae and diagrams. In this chapter, the algorithm of each shading methods are also been discussed and the comparison of shading methods also done.

**8.1 INTRODUCTION**

A software development life cycle model (SDLC) describes the different activities that need to be carried out to developed a software and sequencing of these activities. Mainly a software life cycle is a series of identifiable stage that a software product undergoes during its life time.

We have followed the classical waterfall model of SDLC model, which is shown below in this chapter.

**8.2 CLASSICAL WATERFALL MODEL**

The classical waterfall model is the most obvious way to developed software. We can consider this to be a theoretical way of developing software. It divides the life cycle into six phases given as follows: -

1. Feasibility study
2. Requirement analysis and specification
3. Design
4. Coding
5. Testing
6. Maintenance

**8.2.1 FEASIBILITY STUDY**

The main aim of the feasibility study is to determine whether the project would be financially, technically and environmentally feasible to developed a software or not. Mainly it consists of three steps as follows: -

**An abstract problem definition**

Only the important requirements of the client are captured and details of requirements are ignored.

**Formation of the different strategies for solving the problem**

All the different ways in which the problem can be solved are identified.

**Evaluation of different solution strategies**

The different solution strategies are analyzed to examine their benefits and short comings.

**8.2.2 REQUIREMENT ANALYSIS AND SPECIFICATION**

The aim of the requirement analysis and specification is to understand the exact requirements of the client and to document them properly. This phase consists of two distinct activities as follows: -

**Requirement gathering and analysis**

The goal of requirement gathering is to collect all the information regarding the product to be developed. Whereas the goal of requirement analysis is to weed out the information which is incomplete and inconsistent.

**Requirement specification**

The goal of the requirement specification is to identity the functional requirements, non-functional requirements and the goals of implementation.

**8.2.3 DESIGN**

The goal of the design phase is to transform the requirements specified in the software requirement and specification (SRS) document into a structure that is suitable for implementation in some programming language. Two approaches are used for designing they are as follows: -

**Traditional design approach**

This approach is currently being used by many software development houses. It is based on the data flow-oriented design approach.

**Object oriented design approach**

In this technique various objects that occur in the problem domain and the solution domain are first identified and different relationship that exist among these objects are identified.

**8.2.4 CODING AND UNIT TESTING**

The purpose of coding phase is to translate software design into source code. After coding is done each module of the software is unit tested.

**8.2.5 TESTING**

Testing is the process of determining the current working of each module of a software and then to determine the working status of a software as a whole. Testing is done in three ways they are: -

**Unit testing**

Unit testing involved, testing of each module in isolation from other modules, then debugging it and documenting it. The main objective of unit testing is to determine the current working of individual module.

After coding of each module of this project, we did the unit testing of each phase sequentially.

**Integration testing**

Integration of different module is undertaken once they have been coded and unit tested. During this phase different modules are integrated in a plant manner. Integration of various modules are normally carried out incrementally over a number of steps.

**System testing**

After all the modules have been integrated and tested, system testing is carried out. The goal of the system testing is to ensure that the developed system conforms to its requirements lead out in the SRS document. It consists of three different kind of testing as follows: -

**Alpha testing**

Alpha testing is the system testing performed by the development team. After the development of the software, we the development team performed the alpha testing of the system.

**Beta testing**

It is performed by a friendly set of users. This testing is done by our faculties after the completion of the project.

**Acceptance testing**

This is performed by client itself after the product delivery to determine whether to accept or not. This testing is done by the examiner of our project.

**8.2.6 MAINTENANCE**

Maintenance is the process of improving the functionalities, implementation and to upgrade the software so as to implement the current requirements of the users and to provide the clients more enhanced version of the software. It involves three activities they are: -

**Corrective maintenance**

It involves correcting errors that were not discovered during the product development phase.

**Perfective maintenance**

It involves improving the implementation of the system, enhancing the functionalities according to the customer requirements.

**Adaptive maintenance**

It is required for porting the software to work in a new environment.

**8.2.7** **PHASES OF CLASSICAL WATERFALL MODEL**

Fig.8.2: Classical Waterfall Model

This project has been carried out as per the classical waterfall model of software development life cycle. Because the classical waterfall model is the simplest SDLC model where all the requirements are identified before starting of a project. In our project all the requirements are already identified and documented. So, the selection of classical waterfall model will be the correct choice, because it is simple and straight forward to implement.

**8.3 CONCLUSION**

This chapter summarizes the software development life cycle. In this chapter more emphasis has been given on classical waterfall model of SDLC. This chapter described the different phases of classical waterfall model of SDLC. In this chapter, it is also discussed why this project is based on classical waterfall model of SDLC.

**9.1 INTRODUCTION**

A systemarchitecture or systems architecture is the [conceptual model](https://en.wikipedia.org/wiki/Conceptual_model) that defines the [structure](https://en.wikipedia.org/wiki/Structure), [behavior](https://en.wikipedia.org/wiki/Behavior), and more [views](https://en.wikipedia.org/wiki/View_model) of a [system](https://en.wikipedia.org/wiki/System).An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the [structures](https://en.wikipedia.org/wiki/Structure) and [behaviors](https://en.wikipedia.org/wiki/Behavior) of the system.

A system architecture can consist of system [components](https://en.wikipedia.org/wiki/System) and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called [architecture description languages](https://en.wikipedia.org/wiki/Architecture_description_languages) (ADLs).

**9.2 METHODOLOGY**

This 3D modeling application is divided into four main modules such as

1. Projection
2. Rotation
3. Illumination
4. Shading

To display or to project any object on a computer, we first need to input the real-world coordinates of the vertices of the object like in this project we have taken a cube of as a sample 3D real object whose dimensions height, width and depth are of 20cm. The execution of the program starts by taking the coordinate values of the eight vertices and then convert all the 3D real world coordinates of vertices into 2D coordinates using perspective projection, because computer screen is a 2D plane, so there is no way to plot a 3D coordinate on 2D screen directly but 3D coordinate can be plot on 2D plane after converting from 3D to 2D coordinates. After the conversion of 3D coordinates the software will display a 3D view of that cube.

Once all the 3D coordinates of vertices of the cube have been converted into2D coordinates, the software will need rotation angles about any axis. We can input rotation angles about only one axis or can input rotation angles about all axes at the same time to rotate the cube about all axes. When the rotation had been input, the program will perform conditional operation to check whether the angle is greater than 360 degree or not. If the angle is greater than 360 degree, then different modules like rotation, illumination and shading modules of the software will not be executed that means object will not be rotated and project the same previously projected 3D view of the cube. Otherwise, if the rotation angle is less than 360 degree then the execution of various modules of the software will happen to perform the 3D modelling. That means whenever the rotation angle will be input this whole operation will be executed in a loop.

**9.3** **FLOW CHART OF THE PROPOSED SYSTEM**

A flowchart is a type of diagram that represents an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Though, flowchart is useful in efficient coding, debugging and analysis of a program, drawing flowchart in very complicated in case of complex programs and often ignored.

## **9.3.1 SYMBOLS USED IN FLOWCHART**

Different symbols are used for different states in flowchart, for example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making flowchart.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Purpose** | **Description** |
|  | Flow line | Used to indicate the flow of logic by connecting symbols. |
|  | Terminal(Stop/Start) | Used to represent start and end of flowchart. |
|  | Input/Output | Used for input and output operation. |
|  | Processing | Used for arithmetic operations and data-manipulations. |
|  | Decision | Used to represent the operation in which there are two alternatives, true and false. |
|  | On-page Connector | Used to join different flowline |

Fig. 9.3.1: Symbols used in flowchart

**9.3.2 FLOWCHART OF THE PROPOSED SYSTEM**

Get object coordinates

2D (XY plane) model building using perspective projection

Generate 2D model from 3D model

Perform illumination

Get rotation angle about any axis

Perform shading

Is

Angle >360

Perform rotation

No

Yes

Fig.9.3.2: Flowchart of the system

The execution of the proposed system is exactly same as this flowchart. This flowchart has been described in the methodology section of this chapter.

**9.4 CONCLUSION**

In this chapter, the system architecture of the proposed system has been discussed. The methodology of the proposed system has been discussed as an algorithm and flowchart. This chapter summarizes the basics of a flowchart like the different symbols used in a flowchart.

**10.1 INTRODUCTION**

In this project the six faces of the cube have been colored with six different colors to distinguish and to easily visualize each face of the cube on the computer screen. Here the faces of the cube have been with red, green, blue, black, yellow and pink. As we know that in computer graphics each color is a combination of red, green and blue i.e. RGB color. It means that whatever the color is there, it will be resultant from RGB model. There are many models that [graphic designers](https://www.lifewire.com/what-is-graphic-design-1697521) use to accurately measure and describe color. RGB is among the most important because it is what our computer monitors use to display text and [images](https://www.lifewire.com/adding-images-and-uploading-to-pages-3466470). It is vital that graphic designers understand the difference between RGB and CMYK as well as the working spaces like sRGB and Adobe RGB. These will determine how the viewer sees the colored object in this project. The RGB color model is based on the theory that all visible colors can be created using the primary additive colors of red, green, and blue. These colors are known as primary additives because when they are combined in equal amounts, they produce white. When two or three of them are combined in different amounts, other colors are produced.For example, combining red and green in equal amounts creates yellow, green and blue creates cyan, and red and blue creates magenta. These particular formulae [create the CMYK colors](https://www.lifewire.com/about-the-cmyk-color-model-1697460) used in printing.

**10.2 ANALYSIS OF THE OUTPUTS**

As discussed above in the introduction section of this chapter, we now know that to create or to display any color on a computer display, RGB model is being used. For example, to display the red face of the cube, the amount of red color is 255 which is the maximum intensity of a pixel, the amount of green color and blue color are 0 which is the minimum intensity of a pixel. So, to generate different colors using the RGB model, it is needed to change the amount of each color of RGB model. The change of the amount of color is known as intensity of color or intensity of a pixel. To determine the change of color or intensity change of each face color, we have to perform some calculations like distance between camera and vertices of the cube, angle between camera and normal of the faces and angle between source of the light and normal of the faces of the cube. After these calculations have been done, then only intensity changes can determine. To verify whether the color of each face of the cube is changing or not, we have analyzed all the input parameters and corresponding output parameters. We did the manual calculation first and then we verify with the computer-generated output values to be sure that the output values are correct. The analyzed table is showing in the below figure 10.1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sl. No | Camera position | Source position | Angle of rotation | | | Distance between Camera & Vertices | Angle between Camera and Normal of visible faces | Angle between Source and Normal of visible faces | Initial Intensity of each face | | | | | | Intensity change of each face | | | | | |
|  |  |  | x-axis | y-axis | z-axis |  |  |  | Red | Green | Blue | Black | Pink | Yellow | Red | Green | Blue | Black | Pink | Yellow |
| 1 | 40,-25,50 | -50,-10,30 | 0 | 0 | 0 | v1=52.2  v2=65.8  v3=61  v4=73  v5=68.7  v6=79.5  v7=75.7  v8=85.6 | -0.64  0.28  -0.47 | 74.4  27.7  90.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.04 | -1.42 | -1.15 | 0.69 | 0.85 | 3.01 |
| 2 | 40,-25,50 | -50,-10,30 | 30 | 0 | 0 | v1=57  v2=69.7  v3=55.2  v4=68.2  v5=73.2  v6=83.4  v7=71.8  v8=82.2 | -0.77  -0.45 | -0.45  0.86 | 0 | 0 | 0 | 0 | 0 | 0 | 4 -0.07 | 6 -1.49 | 1 -1.44 | 5 0.69 | 2 0.62 | 3 2.99 |
| 3 | 40,-25,50 | -50,-10,30 | 0 | 45 | 0 | v1=58.3  v2=55.9  v3=66.3  v4=64.2  v5=76.9  v6=75.1  v7=83.2  v8=81.5 | -0.90  0.26 | 0.38  0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.11 | 1.23 | 1.83 | -1.05 | 0.77 | 1.34 |
| 4 | 40,-25,50 | -50,-10,30 | 0 | 0 | 60 | v1=55.9  v2=55.8  v3=69.9  v4=69.9  v5=71.5  v6=71.5  v7=83.0  v8=83.0 | -0.65  0.67 | -0.31  -0.69 | 0 | 0 | 0 | 0 | 0 | 0 | 0.69 | -1.35 | -0.74 | -0.63 | 1.09 | 1.17 |
| 5 | 40,-25,50 | -50,-10,30 | 45 | 37 | 90 | v1=57.8  v2=63.2  v3=58.0  v4=63.8  v5=76.9  v6=81.3  v7=77.7  v8=81.7 | -0.95  -0.27 | 0.19  -0.20 | 0 | 0 | 0 | 0 | 0 | 0 | -1.44 | 0.22 | 1.35 | 0.46 | -0.97 | -0.09 |

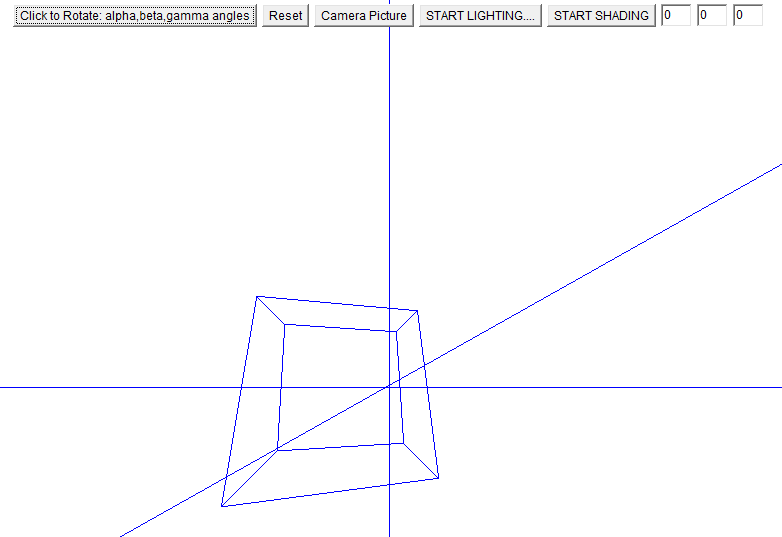
In the above analysis table fig.10.2, second column shows the camera positions and it is fixed in the program as a 3D coordinate value. Second column shows the position of source position, it is also fixed in the program as a 3D coordinate value. Third column is for value of angle of rotation in x-axis, y-axis and z-axis. Fourth column shows the distance between camera & vertices of the cube for each input value of rotation. Fifth column shows the angle between camera and normal of visible faces because we just need the visible faces on the screen. Seventh column shows the angle between source and normal of visible faces. Eight number column is for initial intensity of each face and the last ninth column is for final intensity change of each face color. Here we have shown the analysis for five different input values of rotation. In this project work or software all the values are user defined values, that means user can change each and every value according to their need.

**10.3 GUI DESIGN AND OUTPUT**

GUI is known as Graphical User Interface. GUI is the interface between a user and the software, that provides the platform to user to interact with the software. We have implemented an applet GUI which provides point and click control of software application, eliminating the need to learn a language or type commands in order to run the application.

Here in the GUI, user access is given by which a user can input rotation angles in three different boxes. There are five push buttons provided in the GUI for performing the various operation of the software.

Output of the software has been shown below figures as follows: -

****

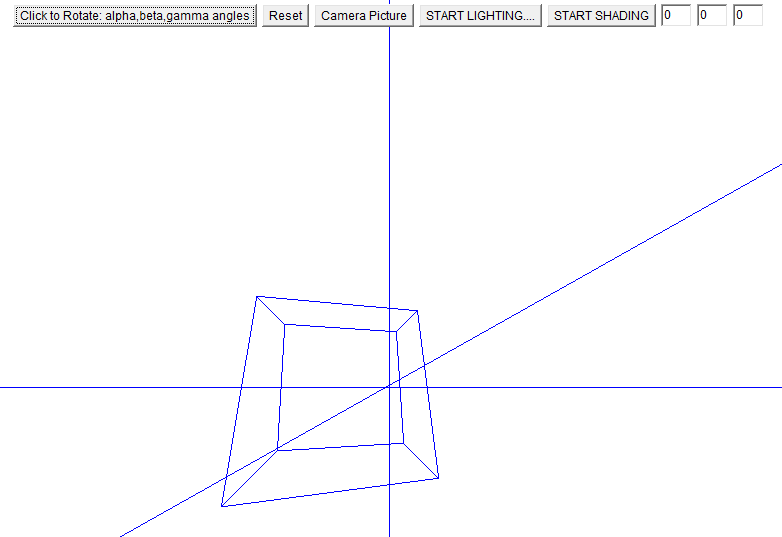
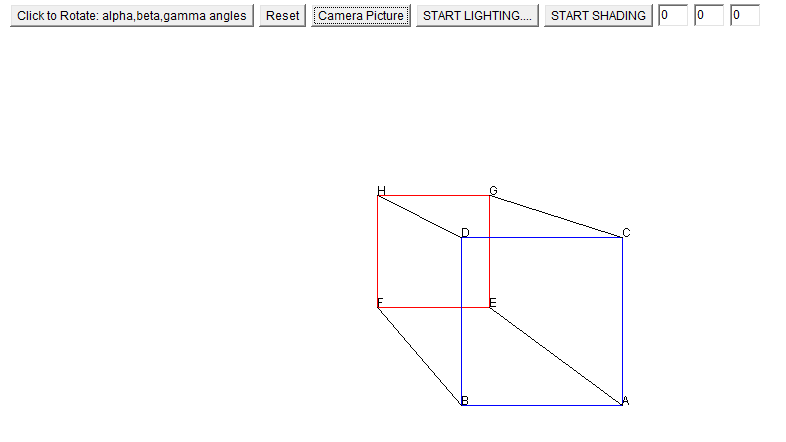
****

Fig. 10.3(a):Normal perspective projection of a cube

Here in this figure 10.3(a) a normal perspective projection of a cube has been shown in which quadrilateral represents the cube and three line segments are representing x-axis, y-axis and z-axis respectively. Here in the GUI first button from left side is for rotation module, when user will click this button then the object will be rotated according to the input rotation values. Second button is for reset, which will reset all change values to its original values. Third button is for camera perspective projection, when the user will click this button, he/she will get a perspective view of a cube as shown in the above figure. Forth button is for illumination i.e. is for lighting effects on the cube. Last button i.e. starts shading button is for color the faces of the cube.

****

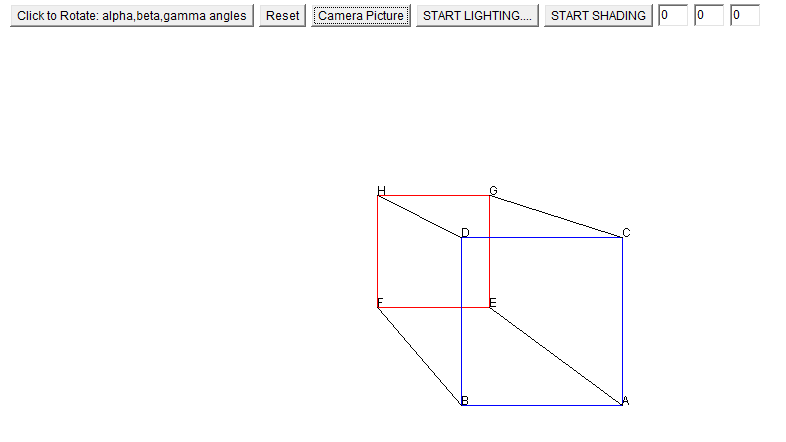
****

Fig. 10.3(b):Camera perspective projection of a cube without rotation

Here in this figure 10.3(b) a camera perspective projection of a cube has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only three faces namely ABCD, HDBF and HGCD faces out of six faces will be visible to the viewer, this scenario can be visualized more accurately and easily when we apply shading as shown in the below figure.

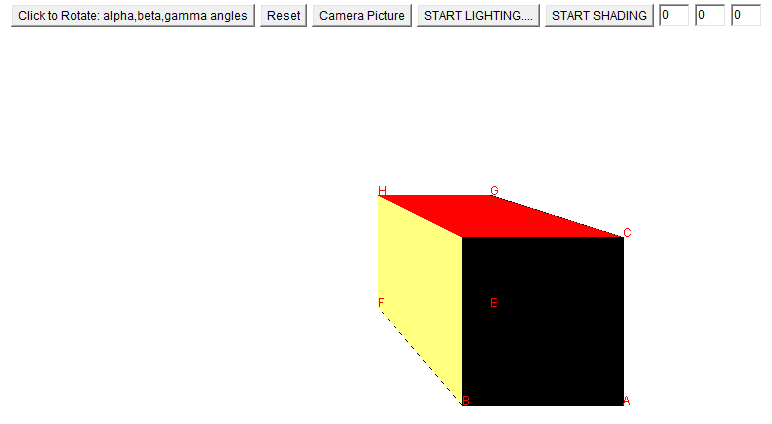
****

Fig. 10.3(c):Camera perspective projection of a cube without rotation with shading

Here in this figure 10.3(c) a camera perspective projection of a cube with shading has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only three faces namely ABCD, HDBF and HGCD faces out of six faces will be visible to the viewer, as we can see that only red, yellow and black faces are visible and others are hide by these faces which is the main purpose of shading in this project.

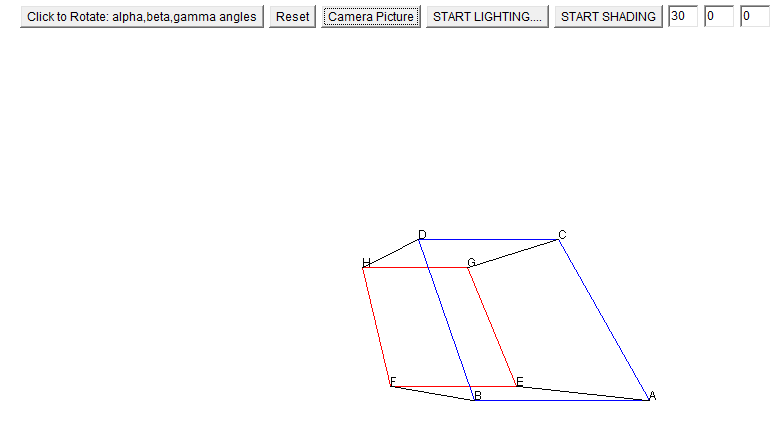
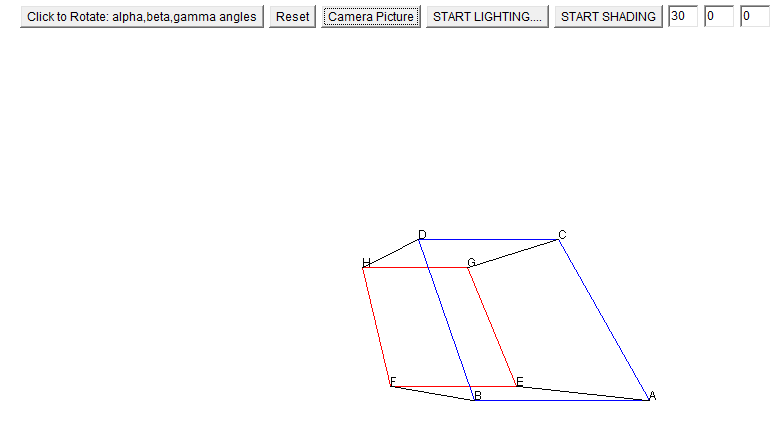
****

Fig. 10.3(d):Camera perspective projection of a cube with rotation about x-axis

Here in this figure 10.3(d) a camera perspective projection of a rotated cube about x-axis at an angle 30 degree has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HDBF faces out of six faces will be visible to the viewer, this scenario can be visualized more accurately and easily when we apply shading as shown in the below figure.

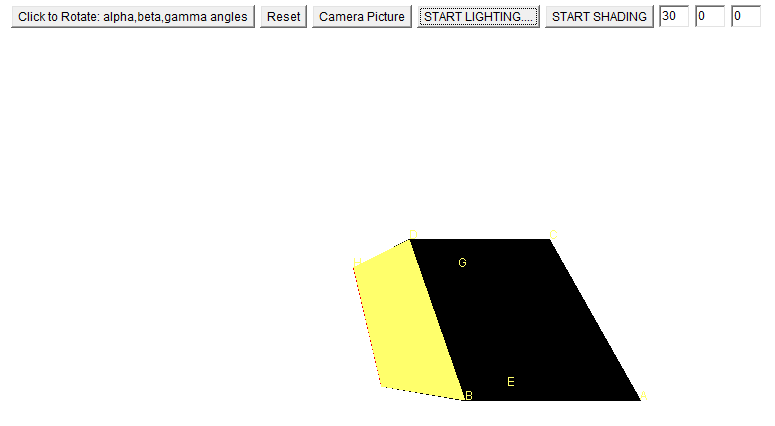
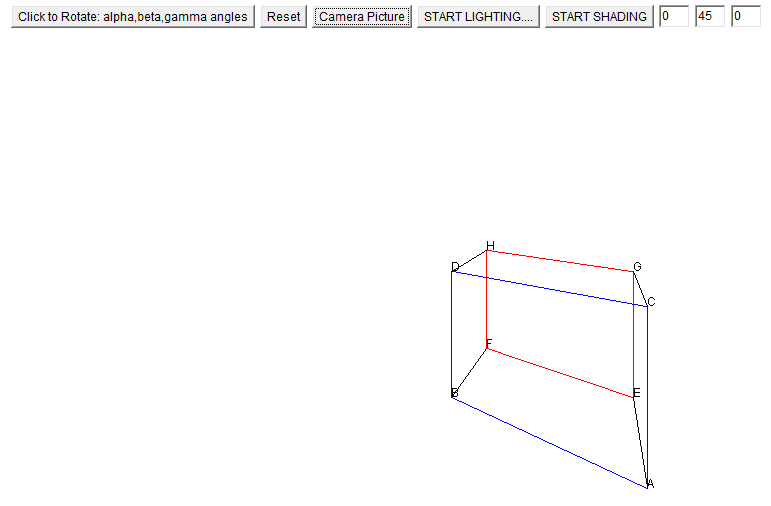
****

Fig. 10.3(e):Camera perspective projection of a cube with rotation and shading

Here in this figure 10.3(e) a camera perspective projection of a rotated cube about x-axis at an angle 30 degree with shading has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HDBF faces out of six faces will be visible to the viewer. As we can see that only yellow and black faces are visible and others are hide by these faces which is the main purpose of shading in this project.

****

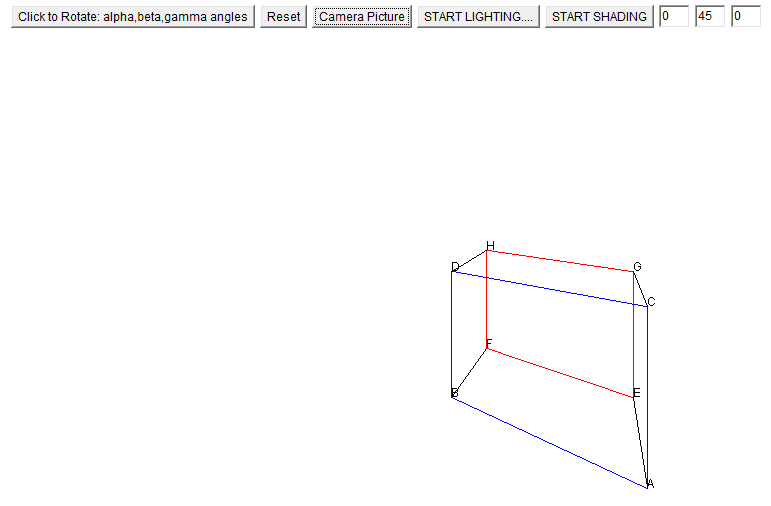
****

Fig. 10.3(f):Camera perspective projection of a cube with rotation about y-axis

Here in this figure 10.3(f) a camera perspective projection of a rotated cube about y-axis at an angle 45 degree has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HGCD faces out of six faces will be visible to the viewer, this scenario can be visualized more accurately and easily when we apply shading as shown in the below figure.

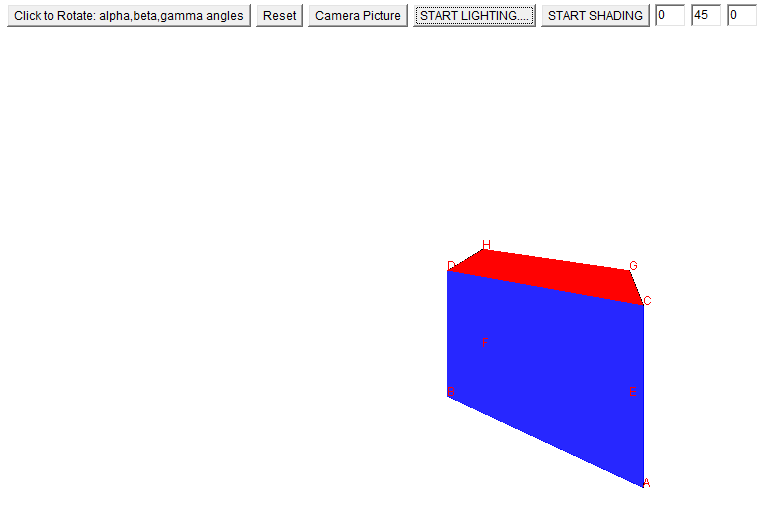
****

Fig. 10.3(g):Camera perspective projection of a cube with rotation and shading about y-axis

Here in this figure 10.3(g) a camera perspective projection of a rotated cube about y-axis at an angle 45 degree with shading has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HGCD faces out of six faces will be visible to the viewer. As we can see that only yellow and black faces are visible and others are hide by these faces which is the main purpose of shading in this project.

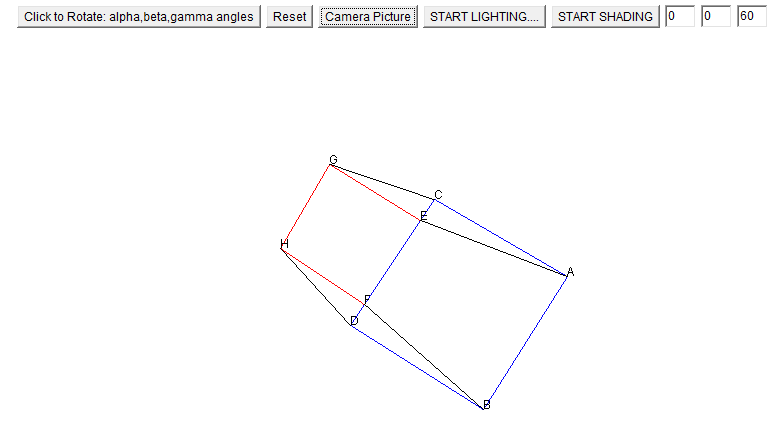
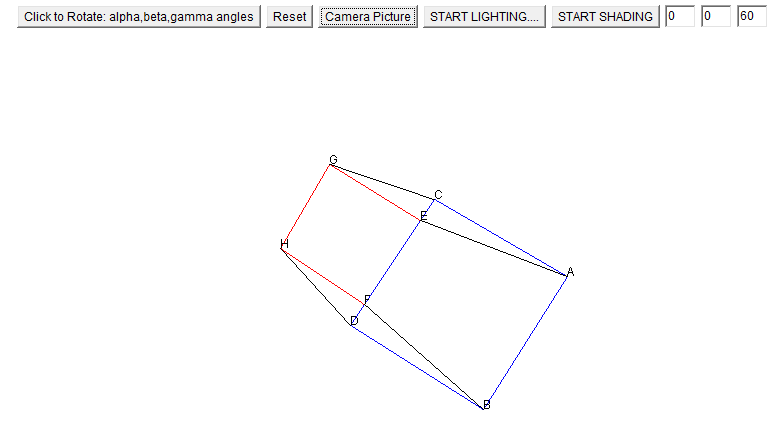
****

Fig. 10.3(h):Camera perspective projection of a cube with rotation about z-axis

Here in this figure 10.3(h) a camera perspective projection of a rotated cube about z-axis at an angle 60 degree has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HGCD faces out of six faces will be visible to the viewer, this scenario can be visualized more accurately and easily when we apply shading as shown in the below figure.

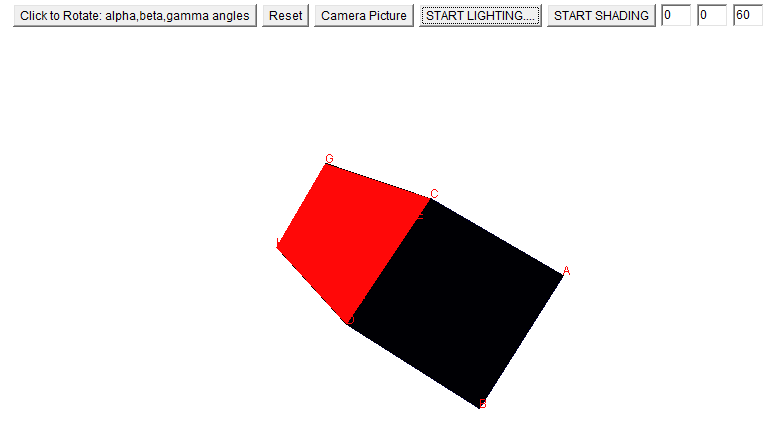
****

Fig. 10.3(i):Camera perspective projection of a cube with rotation and shading about z-axis

Here in this figure 10.3(h) a camera perspective projection of a rotated cube about z-axis at an angle 60 degree with shading has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HGCD faces out of six faces will be visible to the viewer. As we can see that only red and black faces are visible and others are hide by these faces which is the main purpose of shading in this project.

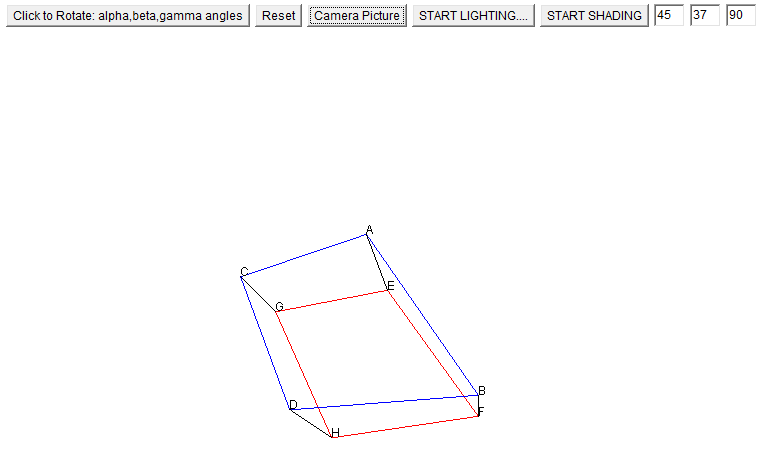
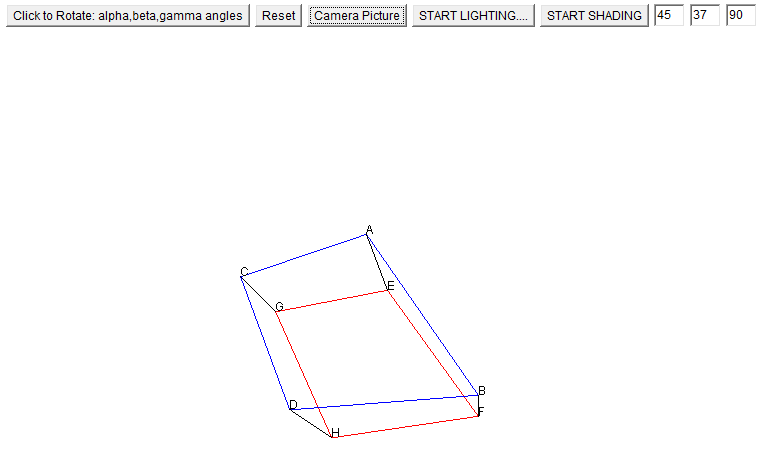
****

Fig. 10.3(j):Camera perspective projection of a cube with rotation about all axis

In this figure 10.3(j) a camera perspective projection of a rotated cube about all axes at an angle 45, 37 and 90 degree has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HDBF faces out of six faces will be visible to the viewer, this scenario can be visualized more accurately and easily when we apply shading as shown in the below figure.

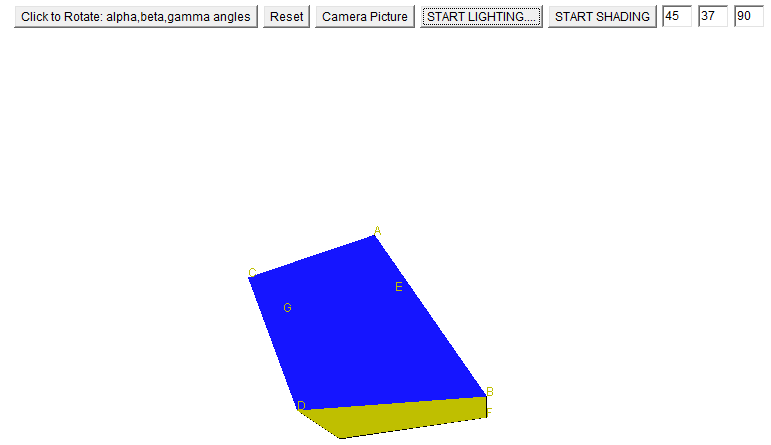
****

Fig. 10.3(k):Camera perspective projection of a cube with rotation and shading about all axis

In this figure 10.3(k) a camera perspective projection of a rotated cube about all axes at an angle 45, 37 and 90 degree has been shown, which can be consider as how the cube is viewing by the viewer. In this projection, only two faces namely ABCD and HDBF faces out of six faces will be visible to the viewer. As we can see that only red and black faces are visible and others are hide by these faces which is the main purpose of shading in this project.

**10.4 CONCLUSION**

This chapter summarizes the different colors used in this project, RGB model, analysis of the results, GUI design and output of the software. In this chapter analysis table of results and screenshots of output of software has been shown and discussed. Here output is shown for five different values of rotation angle.

**11.1 CONCLUSION**

In this paper, we have described the various 3D graphics techniques particularly 3D perspective projection, 3D rotation, 3D translation, 3D scaling, illumination and shading which are used to visualize a 3D object on 2D screen. We have successfully implemented the main phases of the project as mentioned in above and tried to improve the general illumination and shading concept by our own concept. This project is fully mathematical based project. So, we have studied various mathematical theories and concept during the whole project implementation. We have implemented the phases of this project one by one as per the classical waterfall software development life cycle. After implementation of each phase or module, we did the unit testing of software of the software and after successfully testing of each phase, we did the integration testing of the software and finally we did the system testing as one package. The results and outputs of different testing by giving different rotation angle has been shown in the above table of result and discussion section.

**11.2 FUTURE WORK**

In this project we have implemented the flat shading concept which is the basic shading model and not much realistic. So, as a future work this project can be extended by implementing the phong shading model which gives more realistic look of an object than flat shading. This software is meant for desktop so, in future this software can be built for the android system also as the language we used here is java programming language.

**REFERENCES**

1. Computer Graphics: Principles by [John F. Hughes](https://www.amazon.com/John-F.-Hughes/e/B00E5VP9TW/ref=dp_byline_cont_book_1), [Andries van Dam](https://www.amazon.com/s/ref=dp_byline_sr_book_2?ie=UTF8&text=Andries+van+Dam&search-alias=books&field-author=Andries+van+Dam&sort=relevancerank), 2nd edition, [Publisher](https://www.google.co.in/search?rlz=1C1CHBD_enIN796IN796&q=computer+graphics:+principles+and+practice+publisher&stick=H4sIAAAAAAAAAOPgE-LSz9U3MKnKMS-p0lLLKLfST87PyUlNLsnMz9MvL8osKUnNiy_PL8outiooTcrJLM5ILQIAtoZClDYAAAA&sa=X&ved=2ahUKEwjs1p6jg5XfAhXDfX0KHflRCqIQ6BMoADABegQICBAH): [Addison-Wesley](https://www.google.co.in/search?rlz=1C1CHBD_enIN796IN796&q=Addison-Wesley&stick=H4sIAAAAAAAAAOPgE-LSz9U3MKnKMS-pUgKzjZMqjA3ztNQyyq30k_NzclKTSzLz8_TLizJLSlLz4svzi7KLrQpKk3IyizNSiwA8IBHvQgAAAA&sa=X&ved=2ahUKEwjs1p6jg5XfAhXDfX0KHflRCqIQmxMoATABegQICBAI)
2. Computer graphics and Geometric modelling by Solomon D., Publisher: Springer, 1999
3. Computer Animation – Algorithms and Techniques by Parent R., Academic Press, San Diego, 2002.
4. Er.Sheilly Padda, Er.Sonali Gupta, Er.Apoorva Arora, Er. Priya Sharma; “Different Shading Algorithms for Image Processing”; IJARCSSE Volume 4, Issue 5, May 2014; ISSN (online) 2277 128X.
5. Filip Popovski, Igor Nedelkovski, Svetlana Mijakovska; “Generating 3D Model in Virtual Reality and Analysing Performance”; IJCSIT Volume 6, Issue 6, December 2014; ISSN5121-6609.
6. Fundamentals of Computer Graphics by Erik Reinhard, Kelvin Sung, Michael Ashikhmin, 1st Edition,[Publisher](https://www.google.co.in/search?rlz=1C1CHBD_enIN796IN796&q=computer+graphics:+principles+and+practice+publisher&stick=H4sIAAAAAAAAAOPgE-LSz9U3MKnKMS-p0lLLKLfST87PyUlNLsnMz9MvL8osKUnNiy_PL8outiooTcrJLM5ILQIAtoZClDYAAAA&sa=X&ved=2ahUKEwjs1p6jg5XfAhXDfX0KHflRCqIQ6BMoADABegQICBAH): Oxford
7. Garry Bishop and David M. Weimer; “Fast Phong shading”; AT&T Bell Laboratories, Vol 20, Number 4.
8. Georgi Golemshinski1; “Physics Education Using 3D Antenna Simulations”; IJSR Vol 5, Issue 1, January 2016.
9. Glassner; “Situation normal (Gourand and Phong shading)”; IEEE Computer Graphics journal, Vol 17, Issue 2, March 1997, Page 83-87
10. Li Weiying ,JianQiao Coll; “The 3D holographic projection technology based on three-dimensional computer graphics” Audio, Language and Image Processing; (ICALIP), 2012 International Conference.
11. Margaret M. Fleck; “Perspective Projection: The Wrong Imaging Model”; Technical Report 95-01, Department of Computer Science; University of Iowa
12. Mathematics for 3D Game Programming and Computer Graphics by [Eric Lengyel](https://www.amazon.com/Eric-Lengyel/e/B001K8753Q/ref=dp_byline_cont_book_1), 3rd Edition,Publisher: Delmar Cengage Learning.
13. Mr. Suresh T. Gohane, Mr. Rushi N. Longadge; “3D Holograph Projection- Future of Visual Communication”; IJCSN Volume 3, Issue 1, February 2014; ISSN (online)2277-5420.
14. NorhasanaRazzali, MohdShafryMohd Rahim, RoselyKumoi, Daut Daman; “Gouraud Shading to Improve Appearance of Neuronal Morphology Visualization”; International Journal of Biometrics and Bioinformatics (IJBB), Volume 4, Issue 2, January 03, 2011
15. Paul C. Clements (1996); “[A survey of architecture description languages](http://www.sei.cmu.edu/library/assets/Survey_of_ADLs.pdf)”; Proceedings of the 8th international workshop on software specification and design. IEEE Computer Society, 1996.
16. Ron Goldman; “Shading Algorithm”; Department of Computer Science, Rice University.
17. Xi-Dao LUAN, Yu-Xiang XIE, Long YING and Ling-Da WU; “Research and Development of 3D Modeling”; IJCSNS Volume 8, Issue 1, January 2008.
18. [https://www.tutorialspoint.com/computer\_graphics/2d\_transformation.html (10-09-18)](https://www.tutorialspoint.com/computer_graphics/2d_transformation.html%20(10-09-18))
19. <https://www.tutorialspoint.com/computer_graphics/3d_computer_graphics.htm> (10-09-18)
20. <https://www.cs.usfca.edu/~galles/visualization/RotateScale3D.html> (10-09-18)
21. <http://gra-phic.blogspot.com/2011/07/perspective-projections.html>(20-09-18)
22. <https://www.slideshare.net/SanuPhilip/projection-in-computer-graphics> (24-09-18)
23. <https://en.wikipedia.org/wiki/Homogeneous_coordinates> (25-09-18)
24. <https://en.wikipedia.org/wiki/Rotation_matrix> (25-09-18)
25. <http://mathworld.wolfram.com/Perspective.html> (01-10-18)
26. <http://csis.pace.edu/~marchese/CG_Rev/Lect10New/cg_l10new.html> (01-10-18)

1. <https://www.google.com/search?q=three+point+perspective+projection+definition&biw=1242&bih=553&source=lnms&sa=X&ved=0ahUKEwiMxNLYlJzfAhWOdn0KHd-XBpEQ_AUICSgA>( 04-10-18)
2. <https://www.merriam-webster.com/dictionary/three-point%20perspective> (04-10-18)
3. <https://www.chegg.com/homework-help/definitions/perspective-projection-3>(10-10-18)
4. <https://en.wikipedia.org/wiki/Systems_architecture> (15-10-18)
5. http://escience.anu.edu.au/lecture/cg/Illumination/phongShading.en.html (20-10-18) .http://onlinemca.com/mca\_course/kurukshetra\_university/semester5/computergraphics/phong\_shading.php (22-10-18)
6. <https://www.programiz.com/article/flowchart-programming>(22-10-18)