**CHAPTER 1**

**INTRODUCTION**

**1.1 INTRODUCTION TO THE PROJECT**

Every child likes doll to play with it and engaging in various activities in entertainment. Every adult used to have as a decorative item in their collections. In this project the Dancing doll’s are drawn using C/C++ openGL programming using GLUT. The visualization of toys or dolls would be useful in making games with storyline. This project will helps in making model of Dolls and display them on the screen.

In this project the Dolls looks like simple good hence not have a complex attire for it. The doll will have head,without hair- simply bald. It will have coat and pants as cloths and black shoes. The coat will be simple with few buttons fully closed.The Doll should look straight forward. The eye would be more opened like an egg shaped and eyebrow to be crossed and it is neither parallel nor curve.

In openGL programming using GLUT, the object Doll is coded as head, eyes, eyebrow, mouth, teeth, neckring, hands, arms and legs. Both left eye, right eye, left hand, right hand, right leg, left leg, left arm, right arm are coded separately.All objects are drawn using transformation matrix.

Head is coded using the primitive gluSphere(gluNewQuadric()). The eyes are shaped like an egg and a dot(pupil) in it using the function gluSphere(gluNewQuadric()). The middle part consist of neckring, coat and buttons. The neckring is drawn using glutSolidTorus() function. The belly coats are drawn using gluCylinder(gluNewQuadric()) and the transformation matrix and belly bottom is drawn using gluDisk(gluNewQudric()) with matrix. Last is shoes which are covered by gluSphere(gluNewQuadric()) in footright() and footleft() functions.

The function SetupRend() will render the object to make it look more realistic. It helps in enabling depth testing, lighting- Ambient, Diffuse and Specular. One of the important part of openGL programming using GLUT is the user interactions. Here we have both keyboard as well as mouse interactions.

# 1.2 ABOUT OPENGL

OpenGL is a premier environment for developing portable, interactive 2D and 3D graphics applications. Since it introduced in 1992, OpenGL has become the industry’s most widely used and supported 2D and 3D graphics applications programming interface(API), bringing the thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows. OpenGL is callable from Ada, C, C++, Fortran, Python and Java and offers complete independence from network protocols and topologies.

The project Dancing Doll will be designed to access OpenGL directly through functions in 3 libraries namely: GL, GLU, GLUT.

OpenGL

application

program

GLU

Xlib,

Xtk

Frame

buffer

GL

GL

UT

GL

X

Fig 1.1. OpenGL Library tools

# 1.3 PROBLEM STATEMENT

The problem is mainly concerned with modeling a Doll and highlighting some of its unique features and qualities. We have limited ourselves to visualizing and not generating sounds or user interaction as in a game. The doll is subjected to many OpenGL transformations to bring out certain effects like waving or walking.

OpenGL is a software interface to graphics hardware. This interface consists of distinct commands, which we use to specify the objects and operations needed to produce interactive applications. It can be used to implement several visual effects. Hence this project has become more colorful because of the use of the versatile, robust and useful library called OpenGL.

**1.4 OBJECTIVES OF THE PROJECT**

* This project aims to draw a four different colorful 3D dolls using OpenGL libraries, user defined functions in C/C++.
* This is a crude implementation of rotating the scene.
* By using keyboard and mouse interactions we can make the dolls dance.
* This project implements the 3D transformations of objects.
* It is the virtual representation of the objects.

**CHAPTER 2**

**SYSTEM DESIGN**

START

Press ‘B’ to move

Press ‘D’ to move

Display the Dolls and use keyboard function keys to interact

If key==F8??

T

Doll Pinku reacts

Is key==F9??

T

Doll Rinku reacts

Is key==F10??

T

Doll Chinku reacts

Is key==F11??

T

Doll Minku reacts

.

If key==→??

T

Dolls walk towards right

If key==←??

T

Dolls walk towards left

If key==↓??

T

Dolls walk nearer to the user

If key==↑??

T

Dolls walk away the user

Use mouse to rotate the dolls

Press Q to quit

STOP

Fig 2.1. flowchart for the working of dancing dolls

**CHAPTER 3**

**HARDWARE AND SOFTWARE REQUIREMENTS**

# 3.1 HARDWARE REQUIREMENTS

The standard output device is assumed to be a Color Monitor. It is quite essential for any graphics package to have this, as provision of color options to the user is must. The mouse is a main input device, has to be functional i.e. used to give input to the project. A keyboard is used for controlling and inputting the data in the form of characters, numbers i.e. to change the user views.

Apart from these requirements the system should possess the minimum hardware requirements as follows:

* **Hard disk**- Sufficient hard disk space and primary memory available for proper working of the packages to execute the program.
* **Processor**- Pentium 3 or higher processor.
* **RAM**- 96 MB or more.
* 120 MB graphic card for better performance.
* Minimum 1.5 GHz of frequency.

**3.2 SOFTWARE REQUIREMENTS**

Minimum software specification as follows:

* **Operating system-** Windows XP
* **Programming language**- C/C++ with OpenGL.
* **Editor-** Microsoft visual C++ 6.0

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 HEADER FILES USED**

Header files is file with extension .h which contains the functions and macro definitions

to be shared between source files.

1. #include<stdio.h>
2. #include<time.h>
3. #include<GL/glut.h>

**4.1.1. #include<stdio.h>**

In program, printf() is a standard output function, and scanf() is standard input function, these are not defined by the programmer. These functions are already defined inside some language libraries, to use these kinds of predefined functions, we have to include this header file.

**4.1.2. #include<time.h>**

The <time.h> header declares the structures tm, which includes at least the following members:

int tm\_sec seconds[0,61]

int tm\_min minutes[0.59]

int tm\_hour hour[0,23]

int tm\_mday day of month[1,31]

int tm\_mon month of year[0,11]

int tm\_year years since 1900

int tm\_wday day of week[0,6] (sunday =0)

int tm\_yday day of year[0,365]

int tm\_isdst daylight savings flag

The value of tm\_isdst is positive, if daylight saving time is in effect, 0 if daylight saving time is not in effect and negative if the information is not available.

This header defines the following symbolic names:

1.NULL- null pointer constant.

2. CLK\_TCK- number of clock ticks per second.

3. CLOCKS\_PER\_SEC- a number used to convert the value returned by the

clock() function into seconds.

**4.1.3. #include<GL/glut.h>**

The OpenGL utility Toolkit(GLUT) is a library of utilities for OpenGL programs, which primarily performs the system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing number of primitives(both solid and wire frame mode) are also provided, including cubes, spheres, and teapot.

Glut.h includes gl.h and glu.h, so they shouldn’t be included in the source file if glut.h is included.

**4.2 DATA TYPES USED**

**4.2.1.Int**

It contains whole numbers that can have both positive and negative values but no decimal values.

**4.2.2. GLint**

Signed 2’s complement binary integer.

**4.2.3. GLfloat**

An IEEE-754 floating-point value.

**4.3 BUILT-IN FUNCTIONS**

**4.3.1. glBegin() and glEnd()**

**C-specification**

void glBegin(GLenum mode)

void glEnd(void)

**Parameters**

Mode: specifies the primitive or primitives that will be created from vertices presented between glBegin and the subsequent glEnd.

Ten symbolic constants are accepted: GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_TRIANGLES, GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN, GL\_QUADS, GL\_QUAD\_STRIP and GL\_POLYGON.

**Description**

Delimits the vertices that define a primitive or group of like primitives. glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted.

**4.3.2. glClear()**

**C specification**

Void glClear(GLbitfield mask);

**Parameters**

Mask that indicate the buffers to be cleared. The 3 Masks are GL\_COLOR\_BUFFER\_BIT, GL\_DEPTH\_BUFFER\_BIT, and GL\_STENCIL\_BUFFER\_BIT.

**Description**

glClear sets the bit-plane areas of the window to values previously selected by glClearColor, glClearDepth, and glClearStencil.

**4.3.3. glClearColor:**

**C specification**

void glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);

**Parameters**

red, green, blue, alpha- specify the red, green, blue, and alpha values used when the color buffers are cleared. The initial values are 0.

**Description**

glClear specifies the red, green, blue, and alpha values are used by glClear to clear the color buffers.

**4.3.4. glEnable() and glDisable()**

**C specification**

void glEnable(GLenum cap);

void glDisable(GLenum cap);

**Parameters**

cap- specifies the symbolic constant indicating a GL capability

**Description**

glEnable and glDisable enables and disables various server side capabilities.

**4.3.5. glFlush()**

**C specification**

void glFlush(void);

**Description**

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerated itself. glFlush empties all these buffers, force execution of GL commands in finite time.

**4.3.6. glLight():**

**C specification**

void **glLightfv**(GLenum light, GLenum pname, const GLfloat \* params);

**Parameters**

Light-Specifies a light. The number of lights depends on the implementation, but at least eight lights are supported.

Pname - Specifies a light source parameter for light. GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR, etc..

Params - Specifies a pointer to the value or values that parameter pname of light source light will be set to.

**Description**

glLight sets the values of individual light source parameters. To enable and disable lighting calculation, call [glEnable](https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glEnable.xml) and [glDisable](https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/glDisable.xml) with argument GL\_LIGHTING. The lighting parameters are

GL\_AMBIENT

params contains four integer or floating-point values that specify the ambient RGBA intensity of the light. The initial ambient light intensity is (0, 0, 0, 1).

GL\_DIFFUSE

params contains four integer or floating-point values that specify the diffuse RGBA intensity of the light. The initial value for GL\_LIGHT0 is (1, 1, 1, 1); for other lights, the initial value is (0, 0, 0, 1).

GL\_SPECULAR

params contains four integer or floating-point values that specify the specular RGBA intensity of the light.

**4.3.7. glLoadIdentity()**

**C specification**

void glLoadIdentity(void);

**Description**

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix() with the identity matrix, 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1.

**4.3.8. glMatrixMode()**

**C specification**

void glMatrixMode(GLenum mode);

**Parameters**

mode- specifies which matrix stack is the target for subsequent matrix operations.

**Description**

glMatrixMode sets the current matrix mode. Mode can assume one of four values:

GL\_MODELVIEW - Applies subsequent matrix operations to the model view matrix stack.

GL\_PROJECTION- Applies subsequent matrix operations to the projection matrix stack.

GL\_TEXTURE - Applies subsequent matrix operations to the texture matrix stack.

GL\_COLOR - Applies subsequent matrix operations to the color matrix stack.

**4.3.9. glOrtho()**

**C specification**

void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble nearVal, GLdouble farVal);

**Parameters**

Left, right- specify the coordinates for the left and right vertical clipping planes. Bottom, top- specify the coordinates for the bottom and top horizontal clipping Planes. nearVal, farVal- specify the distances to the nearer and farther depth clipping planes. These values are negative if the plane is to be behind the viewer.

**Description**

glOrtho describes a transformation that produces a parallel projection.Multiplies the current matrix with an orthographic matrix.

**4.3.10. glRotatef()**

**C specification**

void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

**Parameters**

angle- specifies the angle of rotation, in degrees. x, y, z- specify the x, y and z coordinates of a vector, respectively.

**Description**

glRotate produces a rotation of angle degrees around the vector x y z. The current matrix is multiplied by a rotation matrix with the product replacing the current matrix.

**4.3.11. glTranslatef()**

**C specification**

void glTranslatef(GLfloat x, GLfloat y, GLfloat z);

**Parameters**

x, y, z- specify the x, y, and z coordinates of a translation vector.

**Description**

glTranslate produces a translation by x y z. the current matrix is multiplied by this translation matrix, with the product replacing the current matrix.

**4.3.12. glScalef()**

**C specification**

void glScalef(GLfloat x, GLfloat y, GLfloat z);

**Parameters**

x, y, z- specify scale factors along the x, y, and z axes respectively.

**Description**

glScale produces a non-uniform scaling along the x, y, and z axes. The 3parameters indicate the desired scale factor along each of the 3 axes.The current matrix is multiplied by this scale matrix, and the product replaces, the current matrix.

**4.3.13. glViewport()**

**C specification**

void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);

**parameters**

x, y- specify the lower left corner of the viewport rectangle, in pixels. width, height- specify the width and height of the viewport. When a GL context is first attached to a window, width and height are set to the dimensions of that window.

**Description**

glViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let xnd and ynd be normalized device coordinates. Then the window coordinates xw and yw are computed as follows:

xw=xnd+1 width 2+x

yw=ynd+1 height 2+y

**4.3.14. gluSphere()**

**C specification**

void gluSphere( GLUquadric\* quad, GLdouble radius, GLint slices, GLint stacks);

**Parameters**

Quad- specifies the quadrics of object.Radius- specifies the radius of the sphere.Slices- specifies the number of subdivisions around the z axis(similar to Lines of longitude).Stacks- specifies the number of subdivisions around the z axis(similar to Lines of latitude).

**Description**

gluSphere draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin. The sphere subdivided around the z axis into slices and the along the z axis into stacks.

**4.3.15. gluNewCylinder()**

**c specification**

void gluCylinder(GLUquadric\* quad, GLdouble base, GLdouble top, GLdouble height, GLintslices, GLint stacks);

**Parameters**

Quad- specifies the quadrics of the object(created with gluNewQuadric). Base- specifies the radius of the cylinder at z=0. Top- specifies the radius of the cylinder at z=height. Height- specifies the height of the cylinder.Slices- specifies the number of subdivisions around the z-axis.Stacks- specifies the number of subdivisions around the z-axis.

**Description**

gluCylinder draws a cylinder oriented along the z axis. The base of the cylinder is placed at z=0 and the top at z=height.

**4.3.16. gluDisk()**

**c specification**

void gluDisk(GLUquadric\* quad, GLdouble inner, GLdouble outer, GLint slices, GLint loops);

**Parameters**

Quad- specifies the quadrics of object.Inner - specifies the inner radius of the disk. Outer- specifies the outer radius of the disk. Slices- specifies the number of subdivisions around the z axis. loops- specifies the number of concentric rings about the origin into which the disk is subdivided.

**Description**

gluDisk renders a disk on the z=0 plane. The disk has a radius of outer a contains a concentric circular hole with a radius of inner. If inner is 0, then no hole is generated. The disk is subdivided around the z-axis into slices and also, about the z axis into rings.

**4.3.17. gluPerspective()**

**C specification**

void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear,GLdouble zFar);

**Parameters**

Fovy- specifies the field of view angle, in degrees, in the y direction.Aspect- specifies the aspect ratio that determines the field of view in the x- direction. The aspect ratio is ratio of x to y. zNear- specifies the distance from the viewer to the near clipping plane. zFar- specifies the distance from the viewer to the far clipping plane.

**Description**

gluPerspective specifies a viewing frustum into the world coordinate system.In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport. For example, aspect=2.0 means the viewers angle of view is twice as wide in x as it is in y. if the viewport is twice as wide as it it tall, it displays the image without distortion.Depth buffer precision is affected by the values specified for zNear and zFar. The greater the ratio of zFar to zNear is, the less effective the depth buffer will Be at distinguishing between surfaces that are near each other.

**4.3.18. glutSolidCube()**

**C specification**

void glutSolidCube(GLdouble size);

**Description**

glutSolidCube render a solid cube. The cube is centered at the modeling coordinates origin with the sides of length size.

**4.3.19. glutSolidTorus()**

**C specification**

void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);

**Parameters**

InnerRadius- inner radius of the torus. outerRadius- outer radius of the torus. nsides- number of sides for each radial section.Rings- number f radial divisions for the torus.

**Description**

glutSolidTorus render a torus centered at the modelling coordinates origin whose axis is aligned with the z axis.

**4.3.20. glPushMatrix() and glPopMatrix()**

**C specification**

void glPushMatrix(void);

void glPopMatrix(void);

**description**

glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. i.e, after a glPushMatrix call, the matrix on the top of the stack is identical to the one below it. glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack.

**4.4. USER-DEFINED FUNCTIONS**

**4.4.1. DrawTextXY()**

**Syntax:**

void DrawTextXY(double x,double y,double z,double scale,char \*s)

**Explanation:**

This function is used to write the text on the output screen. This function takes the arguments such as co-ordinates of x-axis, y-axis, z-axis and the text which is to be print.for example, DrawTextXY(-2.5,0.95,0.0,0.001," Bhuvaneshwari B N ") prints the name at the corresponding positions of x, y and z-axis.

**4.4.2. cover()**

**Syntax:**

void cover()

**Explanation:**

This function is used to design the main page of the project. The main page of the project displays the name of the project, the name and USN of the students handled this project, name of the lecturer who guide the students and institute name. In this function the background color is set using glClearColor() and color for the text using glColor3f().

**4.4.3. menu2()**

**Syntax:**

void menu2()

**Explanation:**

This function is used to design the next page that is help page of the project.This is the page helps the user to handle the four dolls with the keyboard interactions. How dolls are working here?, for which doll which is the key assigned and other information. In this function the background color is set using glClearColor() and color for the text using glColor3f().

**4.4.4. eyeright() and eyeleft()**

**Syntax:**

void eyeright()

void eyeleft()

**Explanation:**

This function is used to draw the eyes separately as right eye and left eye. The function eyeright() used to draw the right eye and eyeleft() used to draw the left eye. Using glTranslatef() function specify the co-ordinates for the right eye and left eye. glScalef()is used to Specify the size of the eye and glColor3f() is used to Specify the color of the eye. gluSphere() draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.5. legleft() and legright()**

**Syntax:**

void legleft()

void legright()

**Explanation:**

This function is used to draw the legs separately as right leg and left leg. The function legright() used to draw the right leg and legleft() used to draw the left leg. Using glTranslatef() function specify the co-ordinates for the right leg and left leg. glScalef()is used to Specify the size of the leg. gluCylinder() draws a cylinder oriented along the z axis.

**4.4.6. armleft() and armright()**

**Syntax:**

void armleft()

void armright()

**Explanation:**

This function is used to draw the arms separately as right arm and left arm. The function armright() used to draw the right arm and armleft() used to draw the left arm. Using glTranslatef() function specify the co-ordinates for the right arm and left arm. glScalef()is used to Specify the size of the arm. gluCylinder() draws a cylinder oriented along the z axis.

**4.4.7. handleft() and handright()**

**Syntax:**

void handleft()

void handright()

**Explanation:**

This function is used to draw the hands separately as right hand and left hand. The function handright() used to draw the right hand and handleft() used to draw the left hand. Using glTranslatef() function specify the co-ordinates for the right hand and left hand. glScalef()is used to Specify the size of the hand. gluSphere() draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.8. mouth()**

**Syntax:**

void mouth()

**Explanation:**

This function is used to draw the mouth.Using glTranslatef() function specify the co-ordinates for the mouth. glScalef() is used to Specify the size of the mouth.

**4.4.9. teeth()**

**Syntax:**

void teeth()

**Explanation:**

This function is used to draw the teeth. Teeth are drawn using glutSolidCube() that render a solid cube. The cube is centered at the modeling coordinates origin with the sides of length size.Using glTranslatef() function specify the co-ordinates for the teeth.

**4.4.10. eyebrowleft() and eyebrowright()**

**Syntax:**

void eyebrowleft()

void eyebrowright()

**Explanation:**

This function is used to draw the eyebrows separately as right eyebrow and left eyebrow. The function eyebrowright() used to draw the right eyebrow and eyebrowleft() used to draw the left eyebrow. glRotatef() produces a rotation of angle degrees around the vector x y z and rotate the object with respect to some angle. gluCylinder() draws a cylinder oriented along the z axis.

**4.4.11.neckring()**

**Syntax:**

void neckring()

**Explanation:**

This function is used to draw the neckring. It is drawn using glutSolidTorus() render a torus centered at the modelling coordinates origin whose axis is aligned with the z axis.

**4.4.12. head()**

**Syntax:**

Void head()

**Explanation:**

This function is used to draw the head. It is drawn using gluSphere(), draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.13. hatmain()**

**Syntax:**

Void hatmain()

**Explanation:**

hatmain is drawn using gluSphere(), draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin

**4.4.14. maintopball()**

**Syntax:**

Void maintopball()

**Explanation:**

maintopball is drawn using gluSphere(), draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin. It is placed above the hat, it look likes a ball.

**4.4.15. hatring()**

**Syntax:**

Void hatring()

**Explanation:**

hatring is drawn using glutSolidTorus() render a torus centered at the modelling coordinates origin whose axis is aligned with the z axis.. It is placed above the hat, it look likes a ring.

**4.4.16. footleft() and footright()**

**Syntax:**

void footleft()

void footright()

**Explanation:**

This function is used to draw the foot separately as right foot and left foot. The function footright() used to draw the right foot and footleft() used to draw the left foot. Foots are drawn using gluSphere(). It draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin

**4.4.17. coatline()**

**Syntax:**

void coatline()

**Explanation:**

Coatline is drawn using glutSolidCube() that render a solid cube. The cube is centered at the modeling coordinates origin with the sides of length size

**4.4.18. bellycoatbottom()**

**Syntax:**

void bellycoatbottom()

**Explanation:**

Belly coat bottom is drawn using gluDisk renders a disk on the z=0 plane. The disk has a radius of outer a contains a concentric circular hole with a radius of inner. If inner is 0, then no hole is generated. The disk is subdivided around the z-axis into slices and also, about the z axis into rings.

**4.4.19. bellycoat()**

**Syntax:**

void bellycoat()

**Explanation:**

Belly coat is drawn using gluSphere(). It draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.20. pupilleft() and pupilright()**

**Syntax:**

void pupilleft()

void pupilright()

**Explanation:**

This function is used to draw the pupil separately as right pupil and left pupil. The function pupilright() used to draw the right pupil and pupilleft() used to draw the left pupil. pupils are drawn using gluSphere(). It draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.21. topbutton(), middlebutton() and bottombutton()**

**Syntax:**

void topbutton()

void middlebutton()

void bottombutton()

**Explanation:**

These buttons are drawn on the coat of a doll. The three buttons are named as top button, middle button and bottom button. Buttons are drawn using gluSphere(). It draws sphere(drawn using gluNewQuadric) of the given radius centered around the origin.

**4.4.20. display()**

**Syntax:**

void display()

**Explanation:**

This function draws the four dolls namely pinku, chinku, minku and rinku on the display screen.

**4.4.21. SetupRend()**

**Syntax:**

void SetupRend()

**Explanation:**

This function setup the output by enabling depth testing, lighting and color tracking using glEnable(). The light parameters are enabled using GL\_LIGHT0.

**4.4.22. walk() and NormalKey()**

**Syntax:**

void walk(int key,int x,int y)

void NormalKey(GLubyte key, GLint x, GLint y)

**Explanation:**

This function helps to interact with the user by using special keys, by pressing a special key the dolls will walk along x-axis and pressing some other key(alphabets) takes you to next page.

.

**CHAPTER 5**

**SNAPSHOTS**

**5.1 Main screen**



Fig 5.1. First page of the output

It is the first page appears when the program runs. It tells who are handled this project under whom guidance and from which institute. That is Jayachithra M and Bhuvaneshwari B N does this project under the guidance of Dharanesh Kumar M L from CIT Gubbi.

**5.2 Help Screen**

This is the page helps the user to handle the four dolls in the project with the keyboard interactions. How dolls are working here?, for which doll which is the key assigned and other information.

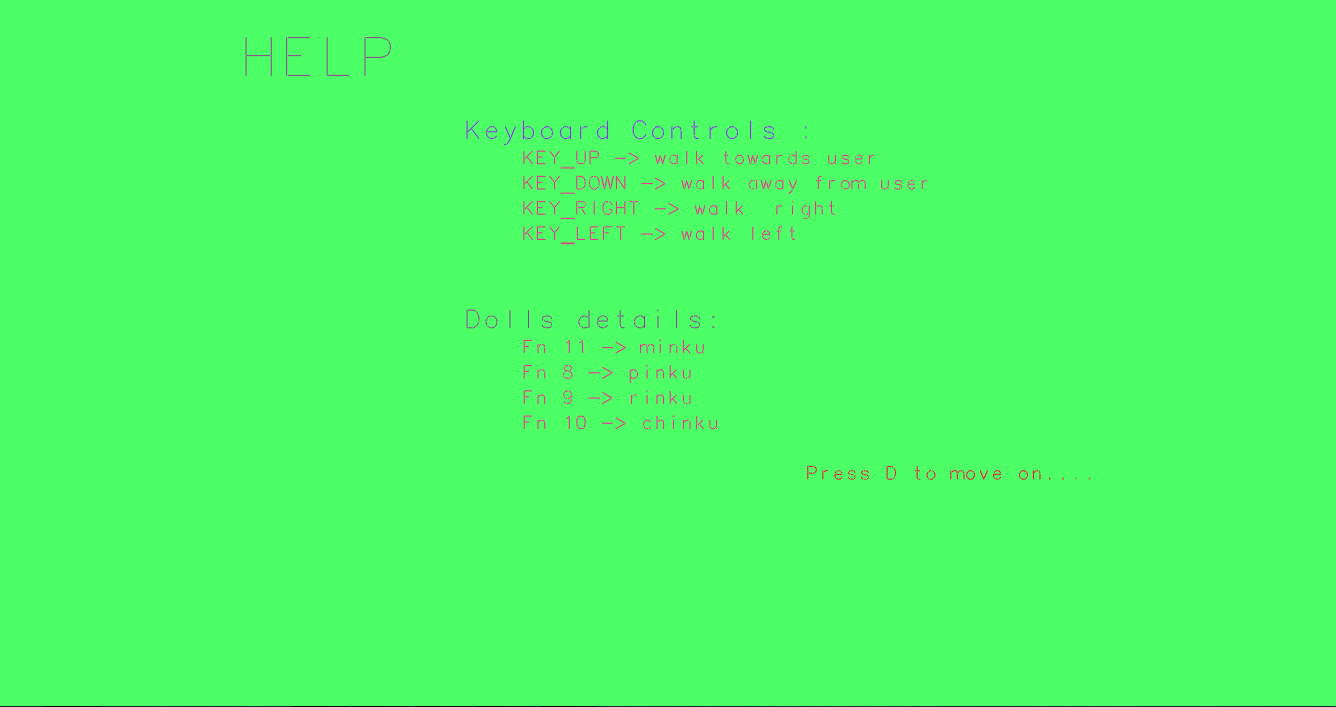


Fig 5.2. Help page

**5.3 The four Dolls(Initially)**

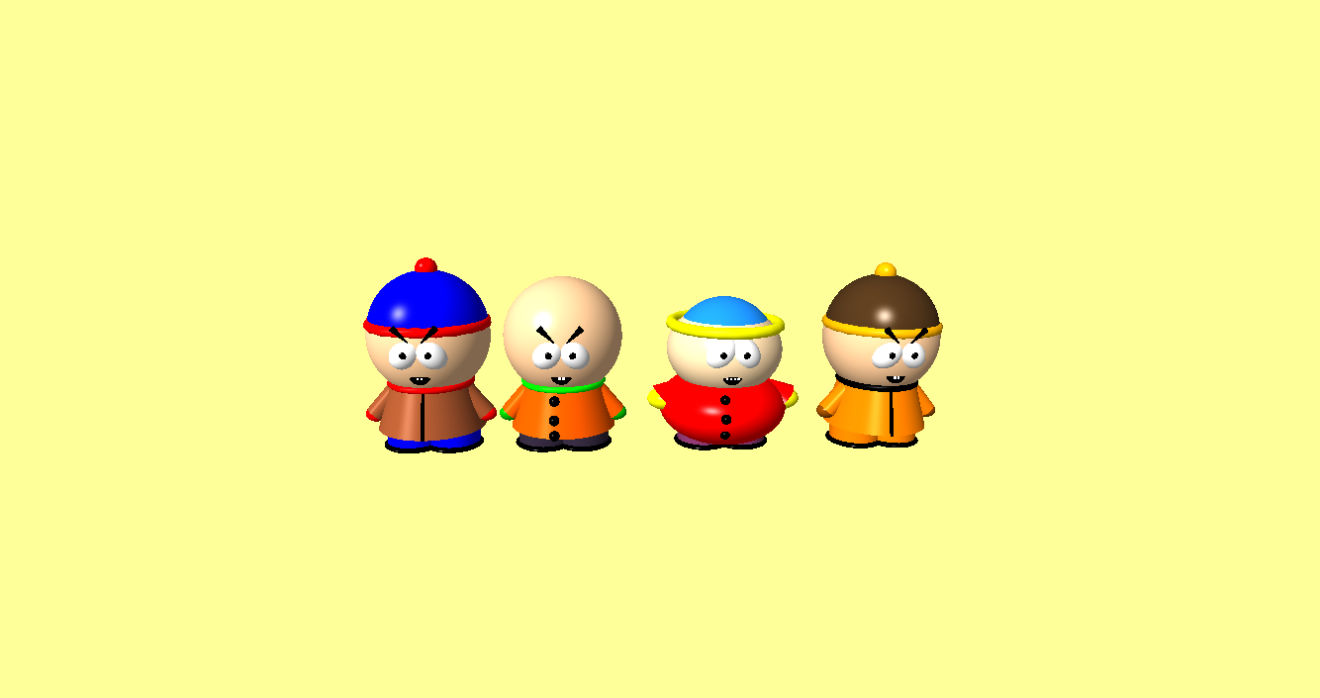


Fig 5.3. The four characters in the project

Initial position of dolls. When the mouse mouse the dolls going to rotate. From left the first dolls is PINKU, second dolls is RINKU, third CHINKU, fourth dolls is MINKU.

**5.4. Result of pressing Fn8**

When user press the function key f8, the doll pinku comes one step forward and remaining dolls remains their respective positions.

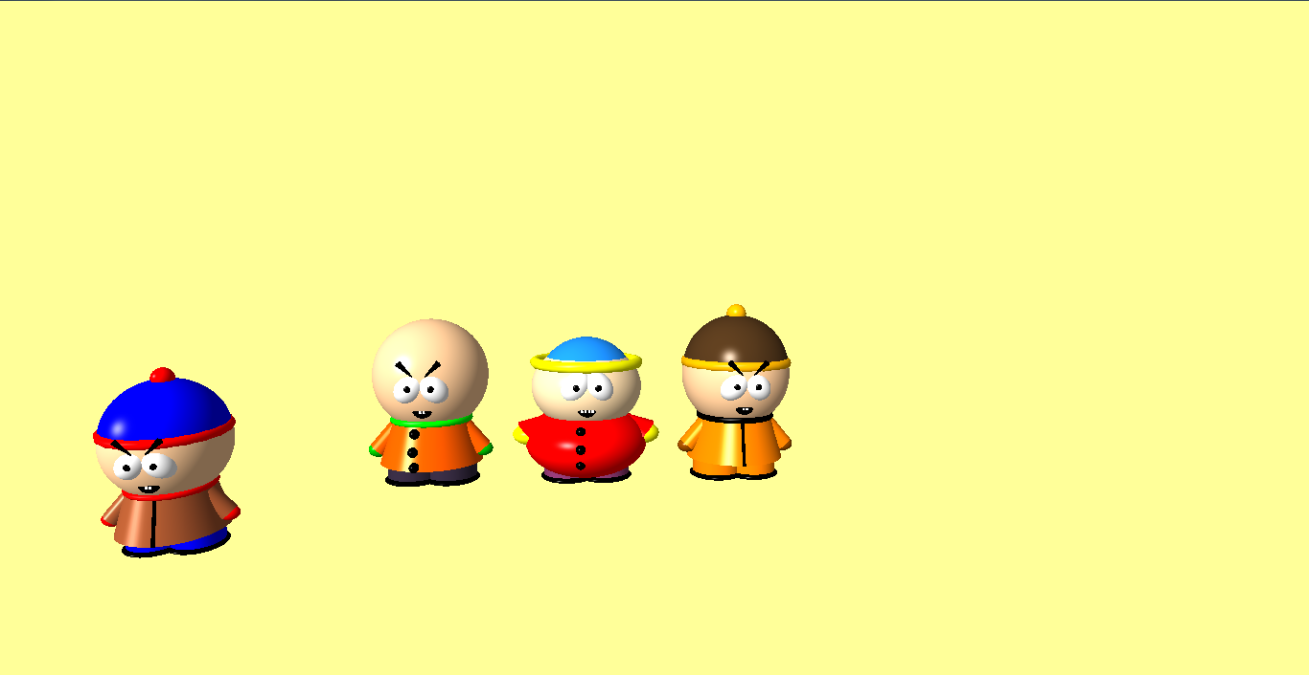


Fig 5.4. Doll pinku comes front

**5.5. zoomed out view of dolls**

when user press the key up button, the dolls become bigger and bigger step by step.

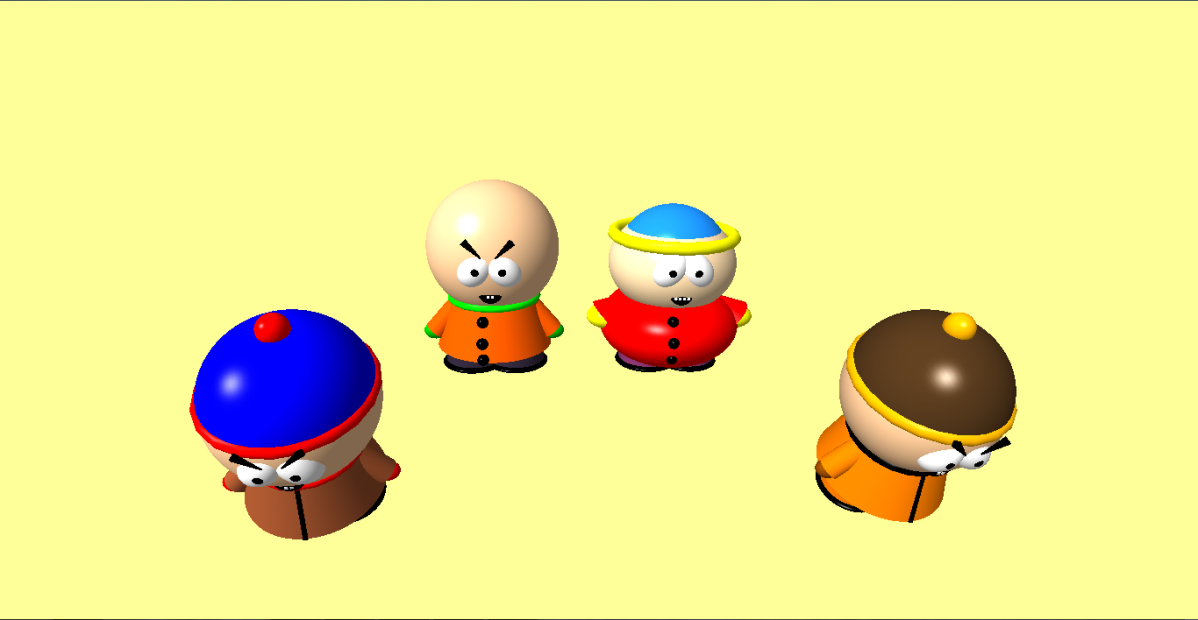


Fig 5.5. zoomed out view of dolls

**5.6 Dolls are rotated when mouse moves.**

As a result of pressing f8, f9, f11 PINKU,RINKU and MINKU comes front and CHINKU remain in its original position. When user press f7 the the dolls positions are paused and the dolls which comes front will rotate in same position with respect to mouse position.

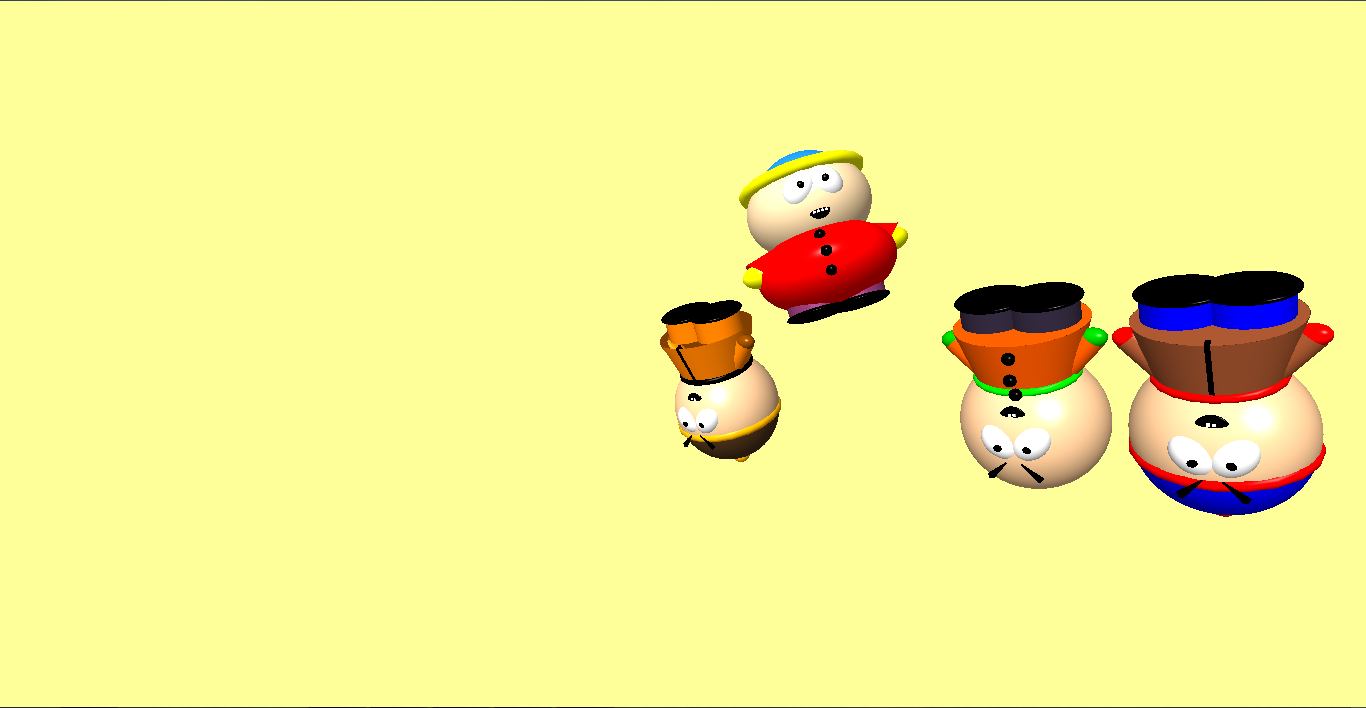


Fig 5.6. The rotating dolls with respect to mouse movement

**CHAPTER 6**

**CONCLUSION**

**6.1 CONCLUSION**

This project has met its objectives to produce a Doll. It has a user friendly interface. It is quite interactive with a good keyboard interface.

The development of this project was very helpful to develop my programming skills and to know c better. It also helped me to implement concepts in my project such as translation, drawing objects on the window screen using points, lines, polygons use of lighting effects, material properties, toggling effects on the objects. This has given me a brief insight as to how programs, involving graphics, are written using OpenGL.

I found designing and developing this PROJECT interesting and a good learning experience. It helped me to learn about computer graphics, design of graphical interfaces, interface to the user, user interaction handling and screen management.

The experience gained during the course of developing this project will prove helpful in future endeavors.

Though the presented project is developed in c, they can be further upgraded to support some other languages as well. Thus the entire source code or the project gives a brief description of the Graphics system concepts being used today (i.e.) the use of OpenGL for various graphics applications or develop interactive graphic models providing 3D view.

I conclude that the project of Doll is completed successfully to the truest of my senses and to best of my abilities.

**REFERENCES**

[1] Angel. E, *OpenGL, A Primer,* Third Edition, Addison-Wesley, Reading, MA, 2008.

[2] Edward Angel, *Interactive Computer Graphics: A Top-Down Approach Using OpenGL, Fifth* Edition,Addison Wesley, 2008.

[3] Francis S. Hill, *Computer Graphics Using Open GL,* Second Edition*,* Prentice Hall, 2000.

[4] Hearn, D., and M.P. Baker, *Computer Graphics,* Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 1994.