VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Minor Project Report on

"DEMONSTRATION OF CANCER CELLS AND ITS CAUSES"

Submitted in partial fulfillment for the Computer Graphics and Visualization Laboratory course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2018-19.

By

Arun M	**4MN17CS005**
Mamatha H S	**4MN17CS014**

|| Under the Guidance of ||

Prof. Bharath Bharadwaj B S

Assistant Professor

Dept. of CS&E

MIT Thandavapura

2019-20

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURANH 766,
Nanjangud Taluk, Mysuru – 571302

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

~~ CERTIFICATE ~~

*Certified that the minor project work entitled "**Demonstration Of Cancer Cells And Its Causes**" is a bonafide work carried out by [**Arun M**] (4MN17CS005) & [**Mamatha H S**] (4MN17CS014) for the Computer Graphics and Visualization Laboratory with course code 15CSL68 of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belagavi during academic year 2018-19.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

Signature of Lab Staff In-Charge

Prof. Bharath Bharadwaj B S

Assistant Professor

Dept. of CS&E

MIT Thandavapura

Signature of the HOD

Prof. Lakshmi Durga

Assistant Professor & HOD

Dept. of CS&E

MIT Thandavapura

External viva

Name of the Examiners **Signature with date**

1)………………………………………………………………………………

2)………………………………………………………………………………

~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration and their whole hearted co-operation all along our project work.

We are grateful to **Dr. Y T Krishnegowda**, principal, MIT Thandavapura, **Dr. H K Chethan,** Professor and Mentor, CS&E, MIT Thandavapura and also **Prof. Lakshmi Durga**, HOD, CS&E, MIT Thandavapura for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and also our friends, who have been a constant source of support and aspirations

Arun M
Mamatha H S

~~~ ABSTRACT ~~~

Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. Not all tumors are cancerous; benign tumors do not spread to other parts of the body Possible signs and symptoms include a lump, abnormal bleeding, prolonged cough, unexplained weight loss and a change in bowel movement. While these symptoms may indicate cancer, they may have other causes. Over 100 cancers affect humans.

In this project, we are demonstrating about cancer cells and its causes. We have shown from basic illustration of cancer to how it spreads, different types of cancer to its causes. It’s a 4 slide basic demonstration of cancer as a disease

In first frame we have showed how cancer is just a uncontrolled division of cells. Cells divide and self destroy when its age is over. There is perfect balance between new born cells and old dying cells. But when it comes to cancer, they divide uncontrollably. This results in uneven shape and not having a proper functionality. Large family of uneven cells will harm the other body tissues and functioning of other organs surrounding it.

In second frame we have shown two main classification of cancer.

- Malignant Tumor
- Benign Tumor

In Malignant tumor are cancerous and are made up of cells that grow out of control. We have shown how cancer cells get into blood stream and spread to other parts of the body affecting other parts or other organs in the body.

In Benign Tumor it forms just one huge mass at a particular site. They do not spread or affect the neighboring cells.S

In third frame is the extension of the second. We have demonstrated the malignant tumor getting into blood stream and developing cancer in other part of the body.

In last frame we have shown the causes of cancer. External factors like alcohol, cigarette, exposure to sun, exposure to asbestos and radioactive material.

~~~~~ CONTENTS ~~~~~

Chapter 1

INTRODUCTION

Computer Graphics become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

Although early application in engineering & science had to rely on expensive & cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Computer Graphics is used in a diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training. Now it can be answered about computer graphics as generalized tool for drawing and creating pictures and simulates the real world situations within a small computer window.

1.1 Aim of the project

The main aim of this project is to a Demonstration of Cancer Cells And Its Causes using OpenGL. The features of the Demonstrating cancer cells is to show how the cells spreads through out blood veins.

1.2 Overview of the project

To make the problem statements better understandable using OpenGL graphics. The main objective of the project is to simplify the understanding of concepts pertaining to the applications of computer science & engineering.

1.3 Outcome of the project

To draw the computer version of the simple ‘Cancer cells and its causes’ and its role in daily life. It is the graphical representation of Cancer cells and it’s causes. It shows how it spreads through our blood veins and which are all the factors that causes cancer.

Chapter 2

DESIGN AND IMPLEMENTATION

2.1Algorithm:

Step 1 : Start

Step 2 :Print the viewing window

Step 3 : Initialize the display window

Step 4 : Intialize the window size

Step 5 : Initialize the window position

Step 6 : Enter the keyboard option for the changes

Step 7 : If key='C' then goto step 9

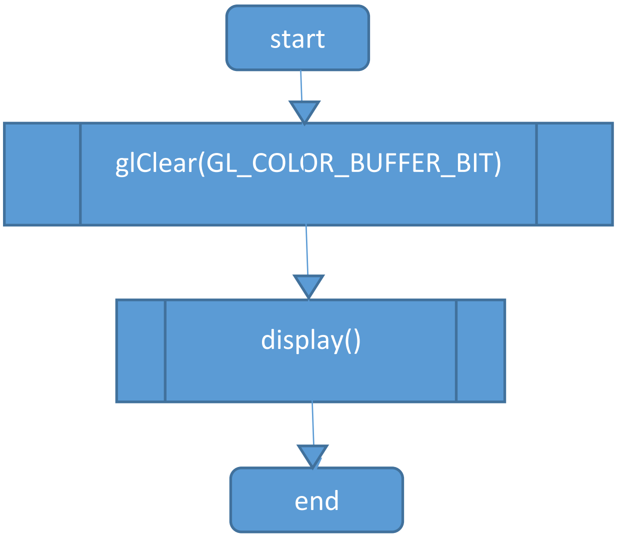
Step 8 : If key='Q' then goto step 10

Step 9 : Display the next frame

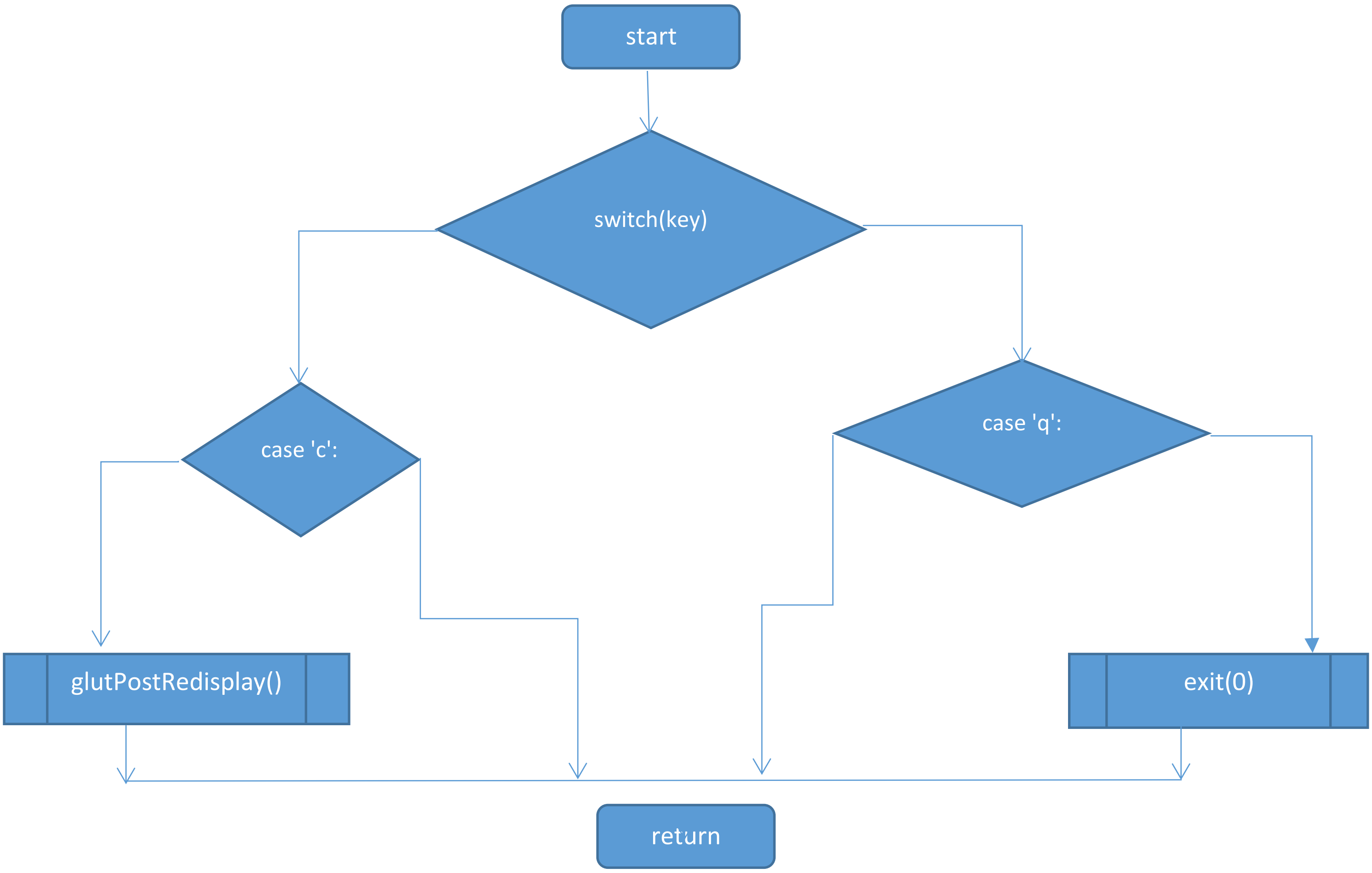
Step 10 : End

2.2 Flowchart:

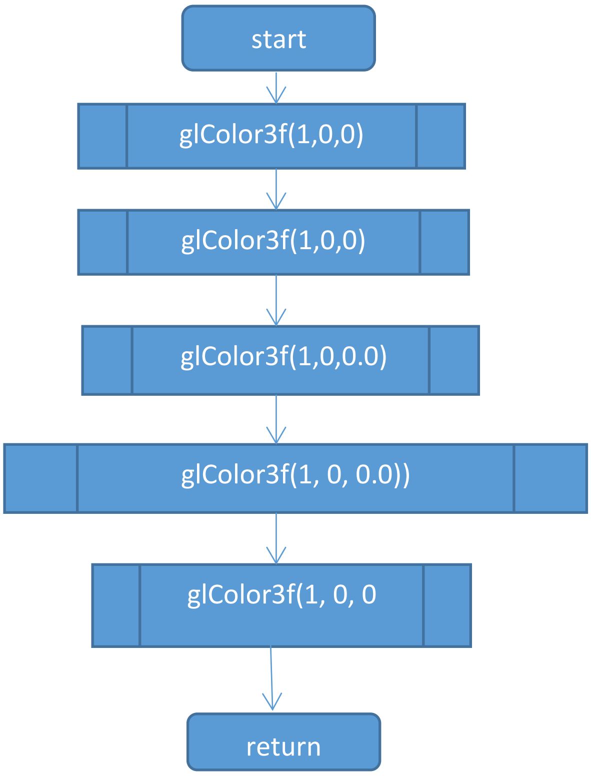
2.2.1 display function:



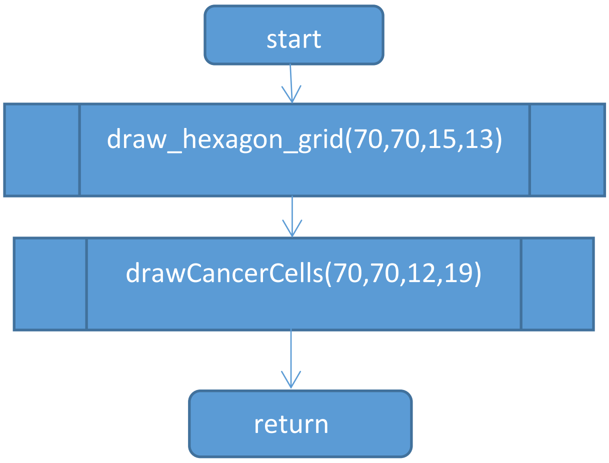
2.2.2 keyboard function



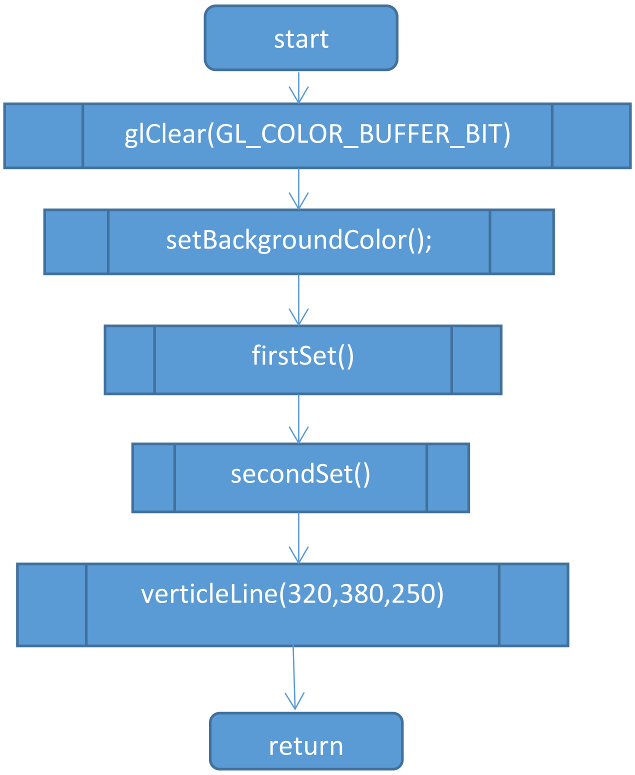
2.2.3 frame0 function



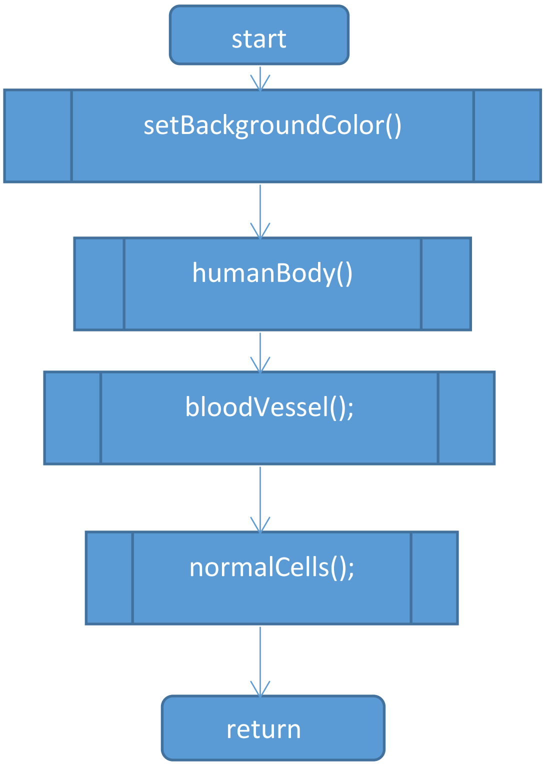
2.2.4frame1 function



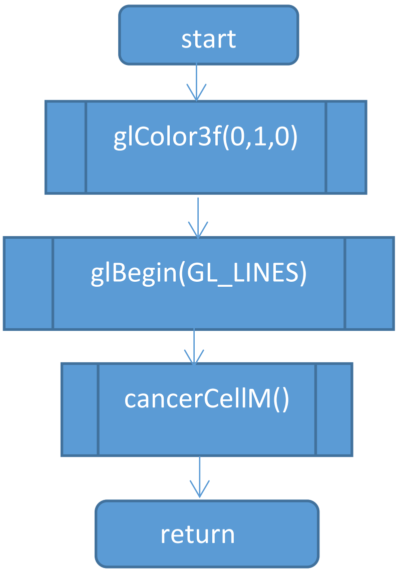
2.2.5 frame2 function



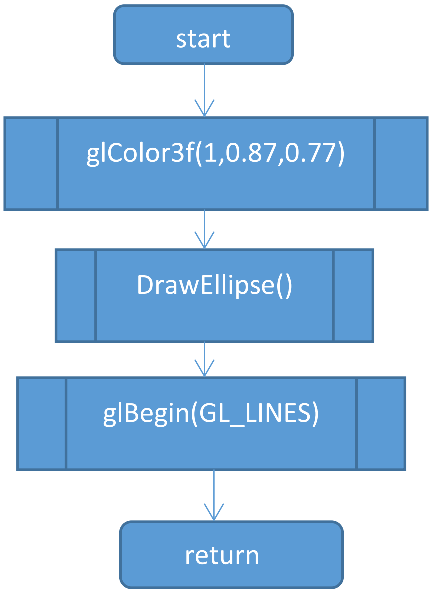
2.2.6 frame3 function



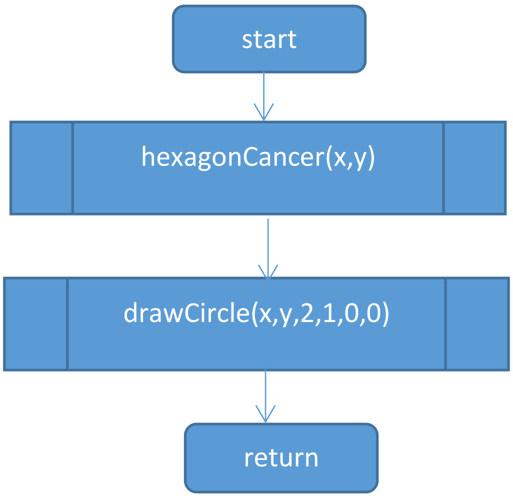
2.2.7 Melignant function



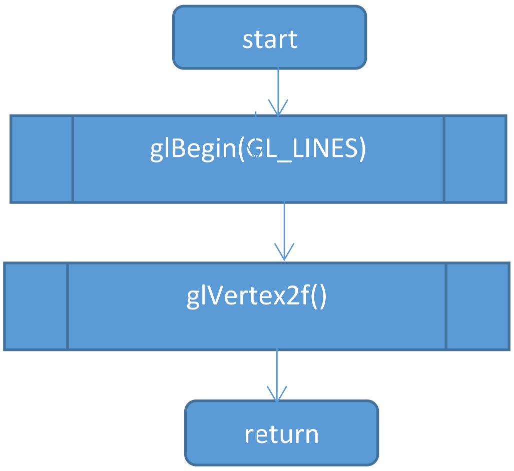
2.2.8 Human function



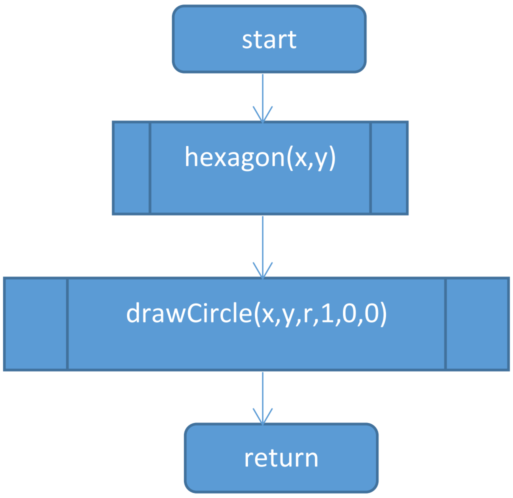
2.2.9 cell cancer function



2.2.10 horzontalLine function



2.2.11 cell function



2.3 OpenGL API's used with description

OpenGL (Open Graphics Library):

OpenGL has become a widely accepted standard for developing graphics application. OpenGL is easy to learn, and it possesses most of the characteristics of other popular graphic system. It is top-down approach. OpenGL is a standard specification defining a cross-language, Cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of different function calls which can be used to draw complex three dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows platforms.

OpenGL provides a powerful but primitive set of rendering command, and all higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

- OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.
- OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, used to hide the complexities of differing window APIs. Rather than using a different library for each system we use available library called openly utility toolkit. It is used as #include <glut.h>.

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows).The second is the **OpenGL Utility Library** (GLU). This Library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

This project makes extensive use of translations, rotations and scaling for creating provides the description of the following functions.

Sl No	OpenGL API's Used	Description
1	**void glBegin(glEnum mode);**	Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES and GL_POLYGON.
2	**void glEnd( );**	It terminates a list of vertices.
3	**void glColor3f[i f d] (TYPE r, TYPE g, TYPE b);**	Sets the present RGB colors. Valid types are int (i), float (f) and double (d). The maximum and minimum values of the floating-point types are 1.0 and 0.0, respectively.
4	**void glClearColor(GLclampf r,GLclampf g,GLclampf b,GLclampf a);**	Sets the present RGBA clear color used when clearing the color buffer. Variables of GLclampf are floating-point numbers between 0.0 and 1.0.
5	**int glutCreateWindow(char *title);**	Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used where there are multiple windows.
6	**void glutInitWindowSize(int width, int height);**	Specifies the initial height and width of the window in pixels.
7	**void glutInitWindowPosition(int x, int y);**	Specifies the initial position of the top-left corner of the window in pixels.
8	**void glutInitDisplayMode(unsigned int mode);**	Request a display with the properties in mode. The value of mode is determined by the logical OR of operation including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE);

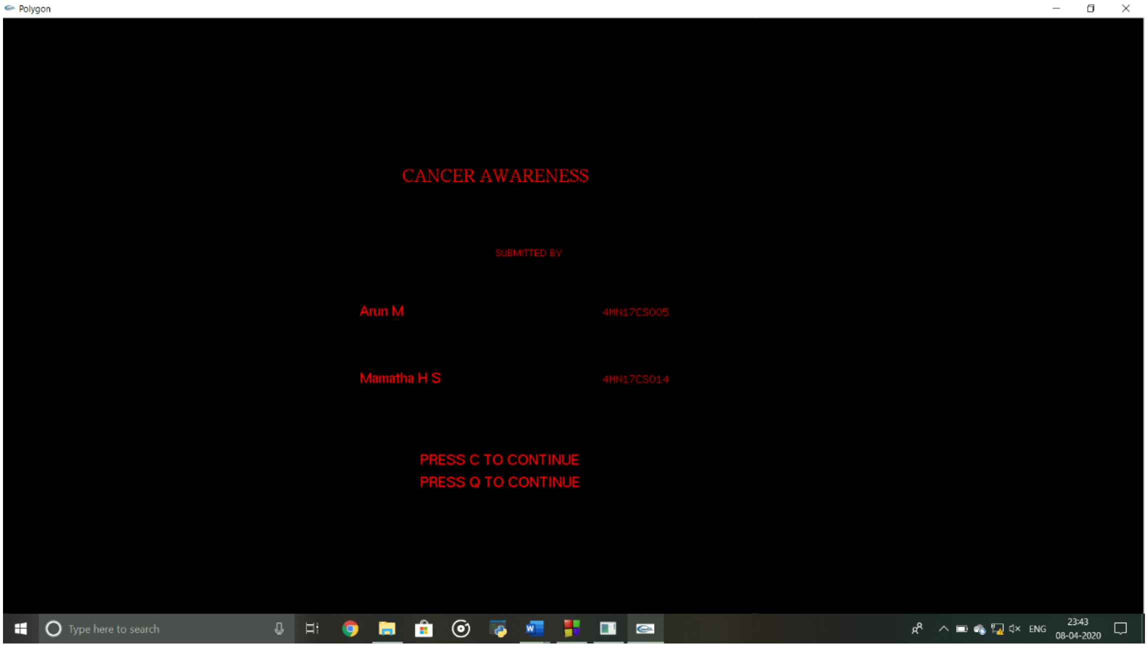
9	**void glFlush( );**	Forces any buffered any OpenGL commands to execute.
10	**void glutInit (int argc, char **argv);**	Initializes GLUT. The arguments from main are passed in and can be used by the application.
11	**void glutMainLoop( );**	Cause the program to enter an event processing loop. It should be the last statement in main.
12	**void glutDisplayFunc(void (*func) (void));**	Registers the display function func that is executed when the window needs to be redrawn.
13	**gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);**	Defines a two-dimensional viewing rectangle in the plane Z=0;
14	**void glutBitmapCharacter(void *font, int char);**	Renders the character with ASCII code char at the current raster position using the raster font given by font. Fonts include GLUT_BITMAP_TIMES_ROMAN_10 and GLUT_BITMAP_TIMES_ROMAN_8_Y_13. The raster position is incremented by the width of the character.
15	**void glClear(GL_COLOR_BUFFER_BIT);**	To make the screen solid and white.
16	**void MouseFunc(myMouse);**	It is used for the implementation of mouse interface. Passing the control to void myMouse(int button,int state,int x,int y);
17	**17 void KeyboardFunc(key);**	It is used for the implementation of keyboard interface. Passing control to void key(unsigned char key,int x,int y);

18	**void glLoadMatrix[fd](TYPE *m);**	Loads the 16 element array of TYPE GLfloat or GLdouble as a current matrix.

Chapter 3

RESULT ANALYSIS

3.1 SNAPSHOTS:



CANCER AWARENESS

SUBMITTED BY

Arun M 4MN17CS005

Mamatha H S 4MN17CS014

PRESS C TO CONTINUE
PRESS Q TO CONTINUE

fig3.1 Introduction frame

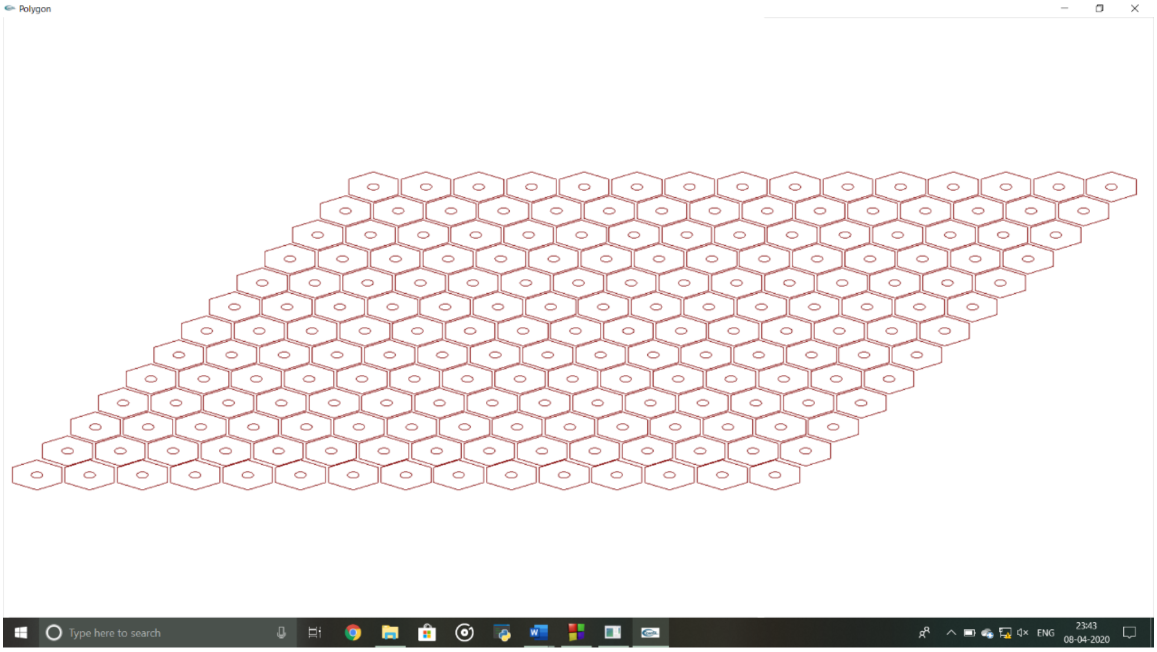


fig3.2 Cell Division

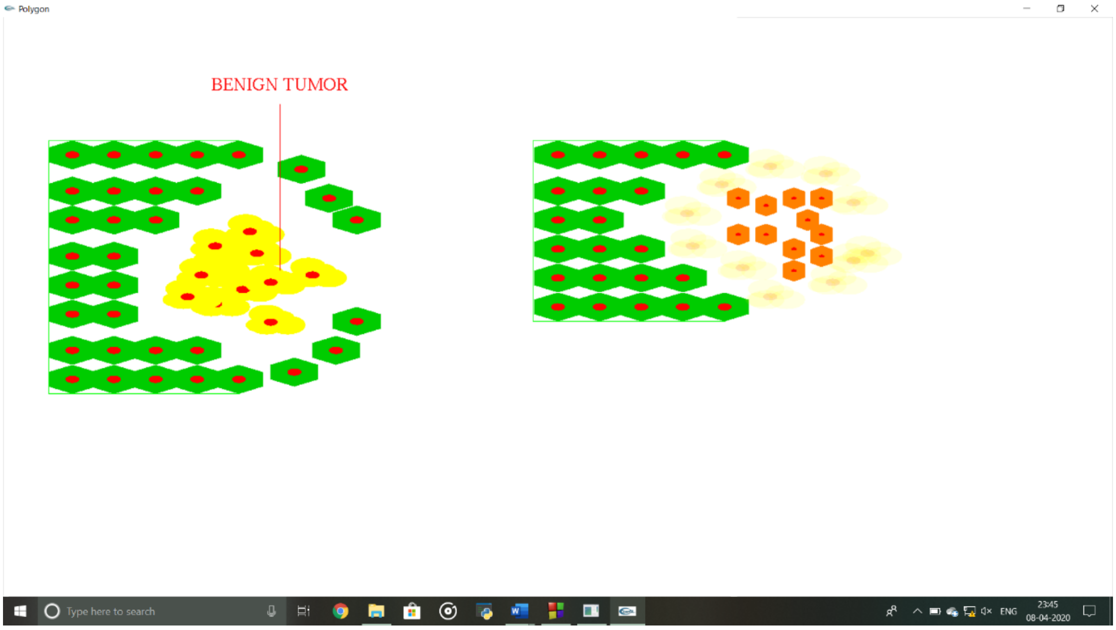


fig3.3 Benign Tumour

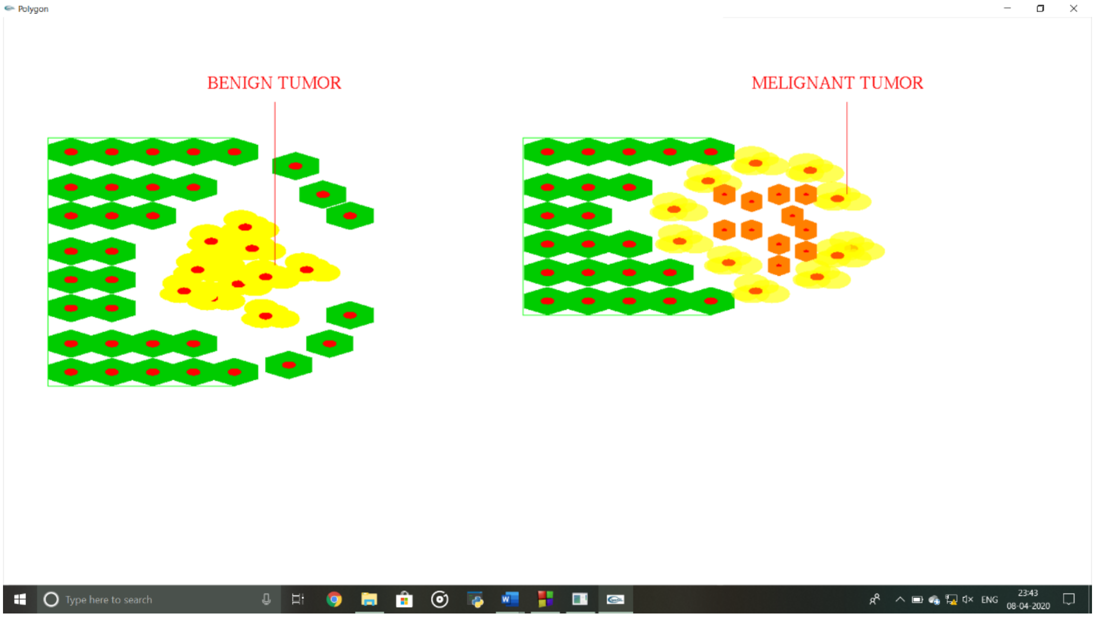


fig3.4 Malignant Tumour

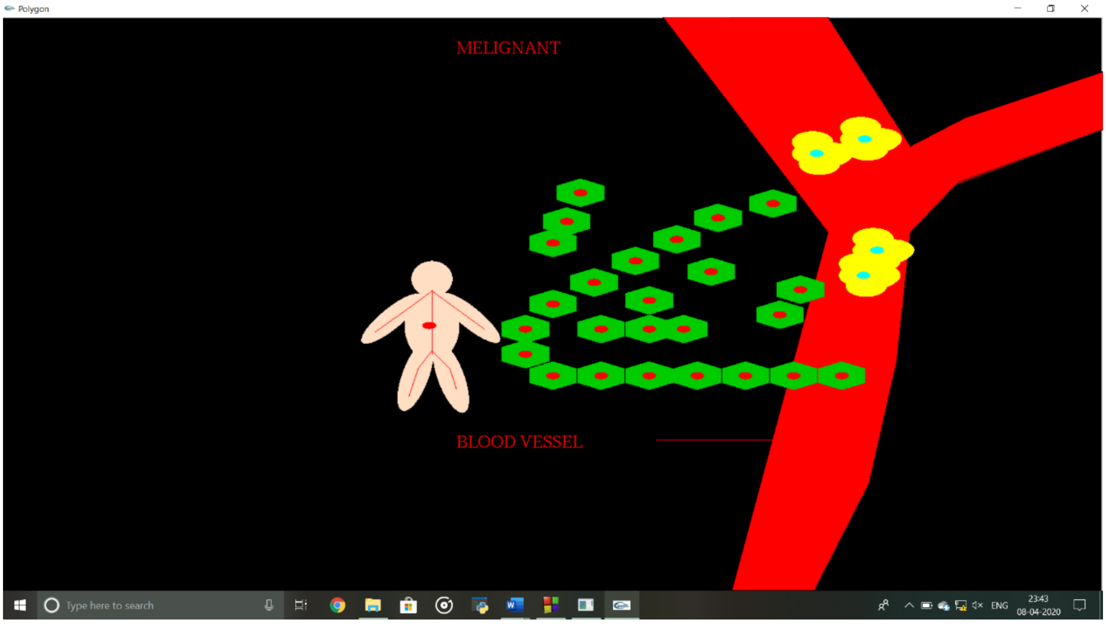


fig3.5 Cancer cells spreading to other part

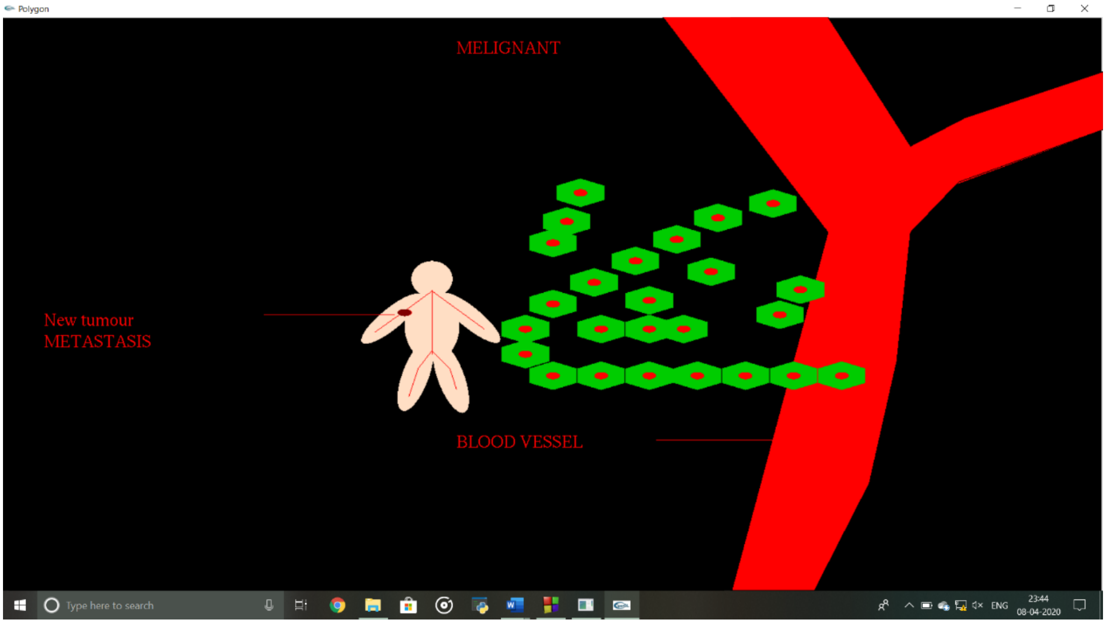
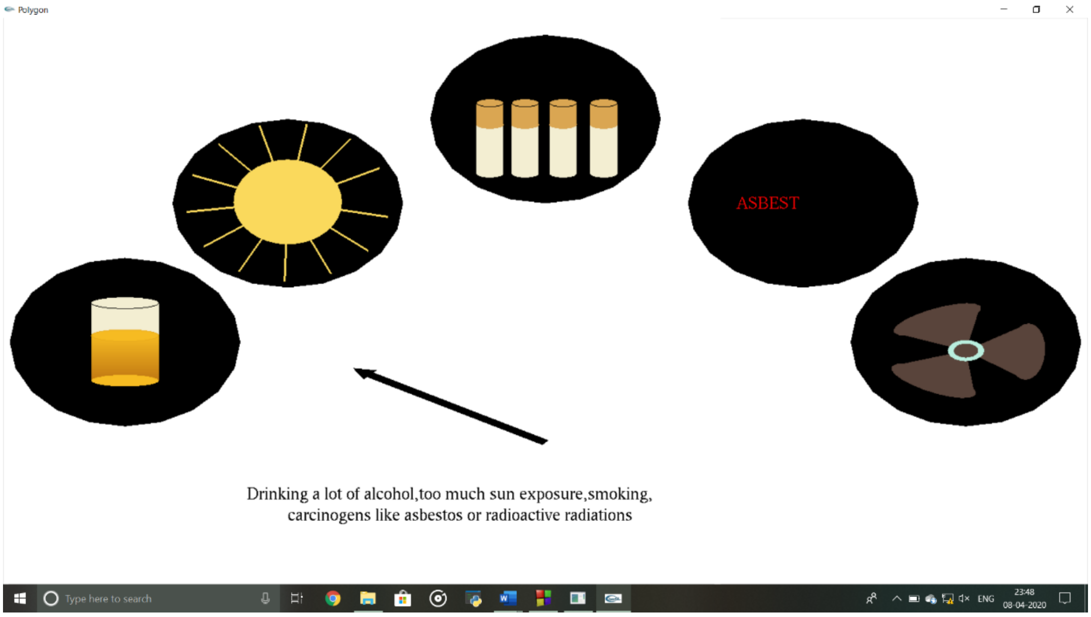
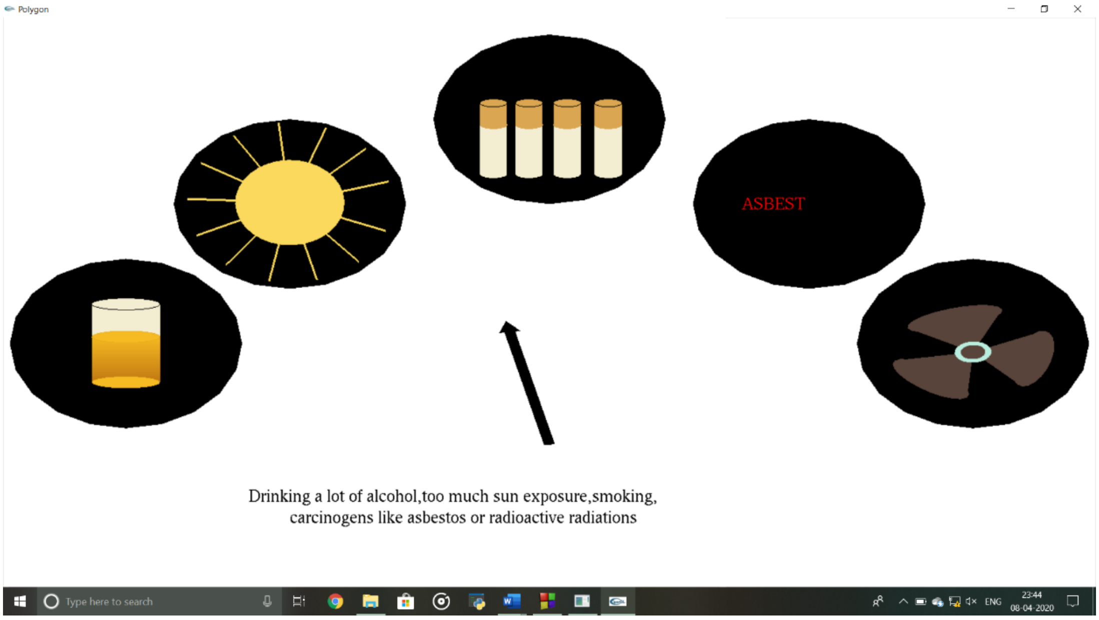


fig3.6 New Tumour Metastasis



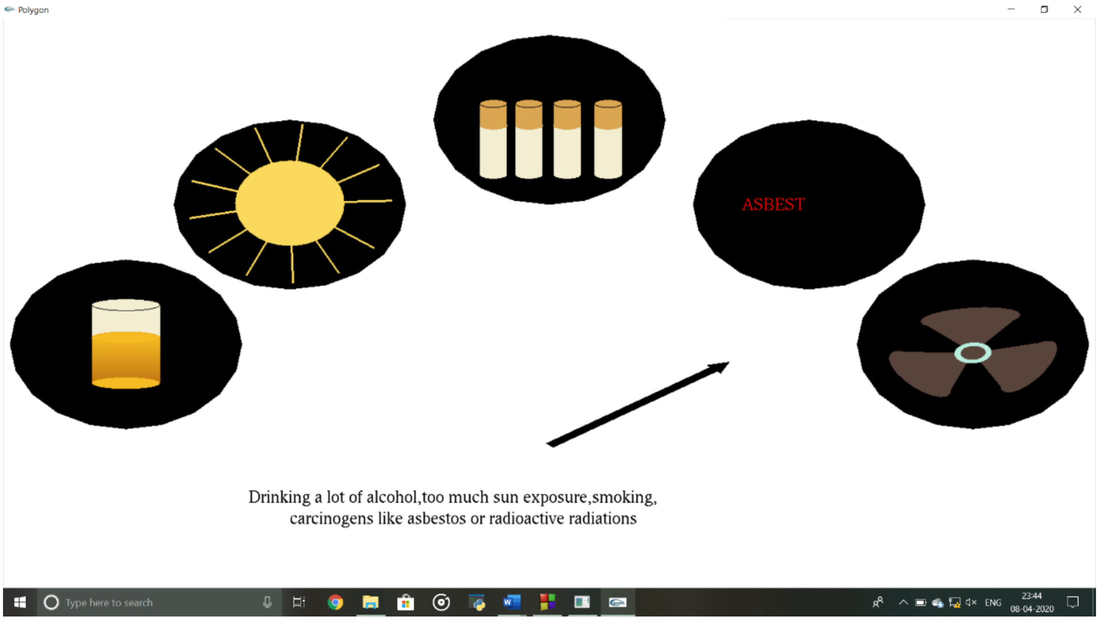
Drinking a lot of alcohol,too much sun exposure,smoking,
carcinogens like asbestos or radioactive radiations

fig3.7 Causes of cancer



Drinking a lot of alcohol,too much sun exposure,smoking,
carcinogens like asbestos or radioactive radiations

fig3.8 Causes of cancer



Drinking a lot of alcohol,too much sun exposure,smoking, carcinogens like asbestos or radioactive radiations

fig3.9 Causes of cancer

3.2Discussion:

1. The program execution starts with the introduction page. The viewer can continue by pressing 'c' key to go to starting frame.
2. On pressing the key, the next page is displayed the first frame.
3. After the animation press c to go next scene of next stage of cancer and its types.
4. Now user can press 'c' for third frame in which in the melignant tumor cells travel in blood vessels which leads in spreading of cancer in other parts of the body.
5. Now press 'c' for next frame to go to last frame which shows us different causes of cancer.
6. Viewer can press 'q' to quit the window.

3.2.1 User Interactions

Demonstration of cancer cells and it’s causes have following user interaction in it-

1. Press 'c' to continue.
2. Press 'q' to quit.

Chapter 4

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

The source code developed is relatively simple, easy to comprehend and works effectively. A simple working model of the famous dining philosophers problem has been implemented. Also the flooding algorithm used in computer networks has been developed to explain the concept with relative ease.

Development of this computer graphics project has given us a better understanding of the topics and concepts learnt in theory. The project increased our knowledge in the field of computer graphics.

The hands on experience of developing a graphics project has helped us inculcate the good practices and principles of software engineering. This project has imbibed within us the spirit of teamwork.

FUTURE ENHANCEMENT:

Even though demo designed is enriched with many options, it is a two dimensional demo, in future it can be redesigned with 3D animation. In future we can make a movement of a 3d cells over blood veins. Making the user interface of this program simpler and more user friendly. In future we can add other types of cancers also

APPENDIX

PROJECT CODE:

```
#include<windows.h>

#include<GL/glut.h>

#include<stdio.h>

#include<stdlib.h>

#include <math.h>

#include<string.h>

#define M_PI 3.14

#define DEG2RAD 3.14159/180.0

float hexagon_r=20;

float op=0;

float hexagon_dx,hexagon_dy,hexagon_gx,hexagon_gy;

#define RADPERDEG 0.0174533

double theta=0;

int frameNumber =0,i;

int f=0,a=0,b=0,c=0,d=0;


void *fonts[] = { GLUT_BITMAP_9_BY_15,

                                    GLUT_BITMAP_TIMES_ROMAN_10,

                                    GLUT_BITMAP_TIMES_ROMAN_24,

                                    GLUT_BITMAP_HELVETICA_18,

            GLUT_BITMAP_HELVETICA_12,
               GLUT_BITMAP_HELVETICA_10 };


typedef struct Point {
  GLfloat x;
  GLfloat y;
} Point;



float yLocation = 0.0f;
float bloby=0,blobx=0,cx1=0,cy1=0,cx2=0,cy2=0;
int c0=0,c1=0,c2=0,c3=0;
void init();
void draw_hexagon(float,float);
void drawCircle(float, float , float , int);
void drawCancerCells(float,float,int,int);

void setBackgroundColor();
void frame0();
void frame1();
void frame2();
void frame3();
void frame4();
void output(int, int ,const char *);
void verticleLine(int ,int ,int );
void hexagonCancer(int ,int);
void drawCircle(GLfloat , GLfloat , GLfloat,int ,int ,int );
void drawCircleCancer(GLfloat , GLfloat , GLfloat,int ,int ,int );
void drawFilledCircle(GLfloat , GLfloat , GLfloat);
void drawHollowCircle(GLfloat , GLfloat , GLfloat,int ,int ,int );
void CellCancer(int,int);
void cell(int,int);
void cell(int ,int ,int );
void cancerCellM(int,int);
void cancerCellB(int,int) ;
void melignant();
void secondSet();
void firstSet();
void humanBody();
void blobAnimation();

void bloodVessel();

void normalCells();

void cellanimation1();
void DrawEllipse(float, float,int,int);
void   hexagon(int,int);
void horzontalLine(int,int,int);
void drawGlass();

void drawCig();

void drawCigs();

void drawSun();

void drawFan();

void Arrow(GLdouble,GLdouble,GLdouble,GLdouble,GLdouble ,GLdouble ,GLdouble );

 void cancerCell(int ,int);




  void drawBitmapText(const char *string, void *font,float x,float y){

    int len, i;

    glRasterPos2f(x, y);

    len = (int)strlen(string);

    for (i = 0; i<len; i++)

    {

      glutBitmapCharacter(font, string[i]);

    }


  }


  void frame0(){

    setBackgroundColor();
    glColor3f(1, 0, 0);

    drawBitmapText( "CANCER AWARENESS", fonts[2],-20, 280);



    glColor3f(1, 0, 0);

    drawBitmapText("SUBMITTED BY", fonts[4],45, 180 );



    glColor3f(1, 0, 0.0);

    drawBitmapText( "Arun M", fonts[3],-50, 100);

    drawBitmapText( "Mamatha H S", fonts[3],-50, 10);



    glColor3f(1, 0, 0.0);

    drawBitmapText( "4MN17CS005", fonts[0],120, 100);

    drawBitmapText( "4MN17CS014", fonts[0],120, 10);



    glColor3f(1, 0, 0);

    drawBitmapText( " PRESS C TO CONTINUE", fonts[3],-10, -100);

    drawBitmapText( " PRESS Q TO CONTINUE", fonts[3],-10, -130);




  }

  void frame2(){

    glClear(GL_COLOR_BUFFER_BIT);

    setBackgroundColor();

    firstSet();
    secondSet();
    output(-150,400,"BENIGN TUMOR");
    verticleLine(-100,380,150);
    if(op<=1)
    {
      op+=0.08;
    }else{
      output(250,400,"MELIGNANT TUMOR");
      verticleLine(320,380,250);
    }
  }
  void frame3(){
    setBackgroundColor();
    humanBody();
    blobAnimation();
    bloodVessel();
    normalCells();
    cellanimation1();
  }


  void setBackgroundColor(){
    glBegin(GL_POLYGON);
    glColor3f(0,0,0);
```

```
   glVertex2f(-499,-499);
   glVertex2f(499,-499);

   glColor3f(0,0,0);
   glVertex2f(499,499);
   glVertex2f(-499,499);
   glEnd();
   glFlush();
  }

void frame4(){
  glColor3f(0, 0, 0);
//  drawGlass(0,0);

  int x, y;
  x=350;
  y=250;
   glColor3f(0, 0, 0);
   //  drawGlass(0,0,0);
   double radius=85;
     glPushMatrix();
         glScalef(1,1.4,1);
   drawFilledCircle(450-x,480-y+25,radius);
   drawFilledCircle(250-x+10,420-y,radius);
   drawFilledCircle(650-x-10,420-y,radius);

   drawFilledCircle(100-x+40,290-y-10,radius);

   drawFilledCircle(800-x-40,290-y-10,radius);

glPopMatrix();

   //drawing glass

   glPushMatrix();

   glTranslated(100-x+40,275-y-10,0);

   glScalef(5,5.5,1);

   drawGlass();

   glPopMatrix();

   //drawing sun

   glPushMatrix();

   glTranslated(250-x+10,420-y+70,0);

       glScalef(1,1.5,1);

   glRotated(-frameNumber*.9,0,0,1);

   drawSun();

   glPopMatrix();




   //drawing cigarette

   glPushMatrix();

   glTranslated(385-x,455-y+100,0);

   glScaled(2,5,1);

   drawCigs();
```

```
glPopMatrix();

//drawing fan

glPushMatrix();

glTranslated(800-x-40,290-y-10,0);

glRotated(-frameNumber*.9,0,0,1);

glScaled(90,100,0);

drawFan();

glPopMatrix();




//print asbest

glPushMatrix();

//glScaled(2,2,0);

glTranslated(600-x-10,410-y+70,0);

glColor3f(1, 0, 0);

drawBitmapText("ASBEST",fonts[2],1,1);

glPopMatrix();






//draw arrow

glPushMatrix();

glScalef(.5,.5,0);

glTranslated(150-x,-(y+70),0);
```

```
   Arrow(400,120,0,90,290,0,8);

   glPopMatrix();



   glColor3f(1, 0, 0);

   drawBitmapText( "Drinking a lot of alcohol,too much sun exposure,smoking,", fonts[2],-
160, -180);

   drawBitmapText( "carcinogens like asbestos or radioactive radiations", fonts[2],-130, -
210);


}

  void bloodVessel()

  {

   //blood vessel cross section start



   glPushMatrix();


   glBegin(GL_POLYGON);


   glColor3f(1,0,0);


   glVertex2f(300,200); //1

   glVertex2f(310,20);
   glVertex2f(270,-150); //2

   glVertex2f(230,-300);


   glVertex2f(290,-300); //3

   glVertex2f(330,-150);


   glVertex2f(350,20); //4

   glVertex2f(360,200);


   glVertex2f(410,300); //5

   glVertex2f(460,330);


   glVertex2f(510,350); //6

   glVertex2f(510,430);


   glVertex2f(400,360); //7

   glVertex2f(360,320);


   glVertex2f(300,500); //8

   glVertex2f(180,500);

   glEnd();


  glPopMatrix();
  glPushMatrix();


  glBegin(GL_LINE_LOOP);


  glColor3f(0,0,0);


  glVertex2f(300,200); //1
 //glVertex2f(310,20);


  //glVertex2f(270,-150); //2
  glVertex2f(230,-300);


  glVertex2f(290,-300); //3
  glVertex2f(330,-150);


  glVertex2f(350,20); //4
  glVertex2f(360,200);


  glVertex2f(395,270); //5
  //glVertex2f(460,330);


  glVertex2f(510,350); //6
  glVertex2f(510,430);
```

```
glVertex2f(400,360); //7

glVertex2f(360,320);


glVertex2f(300,500); //8

glVertex2f(180,500);

glEnd();

glPopMatrix();

glPushMatrix();

glColor3f(1,0,0);

output(30,-100,"BLOOD VESSEL");

horzontalLine(175,300,-90);

glColor3f(1,0,0);

output(30,450,"MELIGNANT");


glPopMatrix();


}


void cell(int x,int y,int r){

  hexagon(x,y);

  drawCircle(x,y,r,1,0,0);

  //drawCircle(x+10,y+5,10,1,0.5,0);

}
```

```
  void init(){

    glEnable(GL_BLEND);

  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    glClearColor(1,1,1,1);



    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-300,500,-300,500);

  }

void keyboard(GLubyte key, GLint x,GLint y){

  switch(key){

    case 'c':f++;

    glutPostRedisplay();break;

     case 'q':exit(0);



     }



}





void initialize(){

  hexagon_dx=hexagon_r*cos(30.0*M_PI/180.0);

  hexagon_dy=hexagon_r*sin(30.0*M_PI/180.0);

  hexagon_gx=2.0*hexagon_dx;
  hexagon_gy=2.0*hexagon_dx*sin(60.0*M_PI/180.0);
  // printf("%d  %d  %d  %d\n",hexagon_dx,hexagon_dy,hexagon_gx,hexagon_gy );
}

void drawFilledCircle(GLfloat x, GLfloat y, GLfloat radius){
  int i;
  int triangleAmount = 20; //# of triangles used to draw circle

  //GLfloat radius = 0.8f; //radius
  GLfloat twicePi = 2.0f * M_PI;

  glBegin(GL_TRIANGLE_FAN);
  glVertex2f(x, y); // center of circle
  for(i = 0; i <= triangleAmount;i++) {
    glVertex2f(
      x + (radius * cos(i *  twicePi / triangleAmount)),
      y + (radius * sin(i * twicePi / triangleAmount))
    );
  }
  glEnd();
}
void draw_hexagon(float x,float y){
  glBegin(GL_LINE_LOOP);
  glVertex2f(x-hexagon_dx,y-hexagon_dy);
  glVertex2f(x-hexagon_dx,y+hexagon_dy);
  glVertex2f(x          ,y+hexagon_r );
  glVertex2f(x+hexagon_dx,y+hexagon_dy);
  glVertex2f(x+hexagon_dx,y-hexagon_dy);
  glVertex2f(x          ,y-hexagon_r );
  glEnd();
  //glFlush();
  drawCircle(x,y,4.0,20);
}

void drawCancerCell(float x,float y,float dx,float dy)
{
//  int r =( rand() % 20)/10.0f;

  glColor3f(0.831f,0.608f,0.627f);
  glBegin(GL_POLYGON);
  glVertex2f(x-hexagon_dx-2*dx,y-hexagon_dy+dy);
  glVertex2f(x-hexagon_dx+dx,y+hexagon_dy+dy);
  glVertex2f(x     +dx     ,y+hexagon_r );
  glVertex2f(x+hexagon_dx-dx,y+hexagon_dy-dy);
  glVertex2f(x+hexagon_dx-dx,y-hexagon_dy+dx);
  glVertex2f(x          ,y-hexagon_r );
  glEnd();
  glColor3f(0.612f,0.220f,0.290f);

  glBegin(GL_LINE_LOOP);

  glVertex2f(x-hexagon_dx-2*dx,y-hexagon_dy+dy);

  glVertex2f(x-hexagon_dx+dx,y+hexagon_dy+dy);

  glVertex2f(x     +dx      ,y+hexagon_r );

  glVertex2f(x+hexagon_dx-dx,y+hexagon_dy-dy);

  glVertex2f(x+hexagon_dx-dx,y-hexagon_dy+dx);

  glVertex2f(x           ,y-hexagon_r );

  glEnd();

  glColor3f(0.596f,0.537f,0.494f);

  drawFilledCircle(x,y,4);

  glFlush();

}

void drawCancerCells(float x,float y,int ni,int nj){

  int i,j; float x0,shiftP=2.0;

  float x1,y1;

  x-=((float)(ni-1))*hexagon_gx*0.5; // just shift x,y to start position (i=0,j=0)

  x-=((float)(nj-1))*hexagon_dx*0.5;

  y-=((float)(nj-1))*hexagon_gy*0.5;

  x1=x+15*hexagon_gx*0.5,y1=y+9*hexagon_gx*0.5;

  glColor3f(1.0,1.0,0.0);


  shiftP+=2;

  /* cancer cells are drawn here
  add amination here*/
  for (x0=x1,j=5; j<nj-2; j++){
    for (i=5; i<ni-2; i++){
      float dx[]= {2,3,1,2,3,4,5,6};
      float  dy[]={7,4,5,3,2,5,7,8,3,2,3};

      drawCancerCell(x1,y1,dx[j-5],dy[j-5]);
      x1+=hexagon_gx+shiftP-.3*i;
    }
    x0+=hexagon_dx+shiftP +.5*i;
    x1=x0+shiftP;
    y1+=hexagon_gy+shiftP;
  }
}/*
void Timer(int iUnused)
{
glutPostRedisplay();
drawCancerCells(-100,-100,10,10);
glutTimerFunc(3000, Timer, 300);
    Timer(10);
}*/

void humanBody(){
  glColor3f(1,0.87,0.77);
```

```
glPushMatrix();

glRotatef(-10,0,0,1);

DrawEllipse(10,50,0,0); //left leg

glPopMatrix();


glPushMatrix();

glRotatef(10,0,0,1);

DrawEllipse(10,50,25,-7); //right leg

glPopMatrix();

DrawEllipse(20,50,12,70); //center torso


glPushMatrix();


glRotatef(60,0,0,1);

DrawEllipse(40,10,60,55); // left arm

glPopMatrix();


glPushMatrix();

glRotatef(-60,0,0,1);

DrawEllipse(40,10,-50,75);// right arm

glPopMatrix();


DrawEllipse(15,25,12,135); //head
```

```
glPushMatrix();


glColor3f(1,0,0);

glBegin(GL_LINES);

glVertex2f(12,30); // center blood vessel

glVertex2f(12,120);


glVertex2f(12,118); // left line

glVertex2f(-8,90);


glVertex2f(-8,90); // left arm

glVertex2f(-30,60);


glVertex2f(12,118); // right vessel

glVertex2f(50,65);


glVertex2f(12,35); // right down  blood vessel

glVertex2f(25,10);



glVertex2f(25,10);  // right down calf blood vessel

glVertex2f(30,-20);


glVertex2f(12,35); // left down blood vessel
```

```
  glVertex2f(2,10);


   // left down blood vessel

  glVertex2f(2,10);

  glVertex2f(-5,-30);


  glEnd();


}


void blobAnimation(){

  //glPopMatrix();

  //glPushMatrix();

  glColor3f(1,0,0);

  glTranslatef(blobx,bloby,0);

  DrawEllipse(5,5,10,35);

  glPopMatrix();


  //update();

  if(c0<78)

  {


    bloby+=5;

    c0+=5;
  }else if(c1<8)
    {
      bloby-=0.7*5;
      blobx-=0.5*5;
      c1++;

  }else{
    glPushMatrix();
    glColor3f(1,0,0);
    output(-270,70,"New TUMOUR");
    output(-270,40,"METASTASIS");
    horzontalLine(-110,-15,85);
    glPopMatrix();
  }
}

void horzontalLine(int Lx1,int Lx2,int Ly){
  glBegin(GL_LINES);
  glVertex2f(Lx1,Ly);
  glVertex2f(Lx2,Ly);
  glEnd();
}
void DrawEllipse(float radiusX, float radiusY,int posx,int posy)
```

```
{
  int i;

  glBegin(GL_POLYGON);

  for(i=0;i<360;i++)
  {
    float rad = i*DEG2RAD;
    glVertex2f(posx+cos(rad)*radiusX,
            posy+sin(rad)*radiusY);
  }

  glEnd();
}

void drawHollowCircle(GLfloat x, GLfloat y, GLfloat radius,int r,int g,int b){
        int i;
        int lineAmount = 100; //# of triangles used to draw circle

        //GLfloat radius = 0.8f; //radius
        GLfloat twicePi = 2.0f * 3.14;
glColor3f(r,g,b);
        glBegin(GL_POLYGON);
                for(i = 0; i <= 300;i++) {
                       glVertex2f(
                           x + (radius * cos(i *  twicePi / lineAmount)),
                           y + (radius* sin(i * twicePi / lineAmount))
                       );
                }
        glEnd();
}
 void cancerCell(int x,int y){
  int r=15,R=1,G=1,B=0;


  drawHollowCircle(x,y,r,R,G,B);
  drawHollowCircle(x+15,y,r,R,G,B);
  drawHollowCircle(x+5,y-15,r,R,G,B);
  drawHollowCircle(x+13,y-3,r,R,G,B);
  drawHollowCircle(x,y+16,r,R,G,B); // nucleus
  drawHollowCircle(x+3,y,5,0,1,1);


 }
void cellanimation1(){
  int x1=0,y1=0,ctr1=0,ctr2=0;


 glPushMatrix();
 glTranslatef(cx1,cy1,0);
 cancerCell(300,100);
 cancerCell(335,120);
 glPopMatrix();

 if(c2<20)
 {
   cy1+=0.5*10;
   cx1+=0.1*10;

   c2++;
 }else if(c2<835){

     cy1+=0.5*15;
     cx1-=0.15*15;
     c2++;
}

 glPushMatrix(); // second set of cells
 glTranslatef(cx2,cy2,0);
 cancerCell(330,35);
 cancerCell(320,0);
 glPopMatrix();


 if(c2<20)
 {
   cy1+=0.5*10;
   cx1+=0.1*10;


   c2++;
 }else if(c2<835){


     cy1+=0.5*15;
     cx1-=0.15*15;
     c2++;
}


     glPopMatrix();


 if(c2<20)
 {
   cy1+=0.5*10;
   cx1+=0.1*10;


   c2++;
 }else if(c2<835){


     cy1+=0.5*15;
     cx1-=0.15*15;
```

```
    c2++;
}


 if(c3<40){
   cy2+=0.5*10;
   cx2+=0.01*10;
   c3++;
 }else if(c3<50){
   cy2+=0.68*10;
   cx2+=0.35*10;
   c3++;
 }else if(c3<700){
   cy2+=0.5*15;
   cx2+=0.70*15;
   c3++;
 }


   if(c3<40){
   cy2+=0.5*10;
   cx2+=0.01*10;
   c3++;
 }else if(c3<50){
   cy2+=0.68*10;
   cx2+=0.35*10;
   c3++;
 }else if(c3<700){
   cy2+=0.5*15;
   cx2+=0.70*15;
   c3++;
 }

 if(c3<40){
   cy2+=0.5*10;
   cx2+=0.01*10;
   c3++;
 }else if(c3<50){
   cy2+=0.68*10;
   cx2+=0.35*10;
   c3++;
 }else if(c3<700){
   cy2+=0.5*15;
   cx2+=0.70*15;
   c3++;
 }

}

void normalCells(){
```

```
cell(80,30,5);

cell(80,65,5);

cell(135,65,5);

cell(170,65,5);

cell(195,65,5);

cell(265,85,5);

cell(280,120,5);


cell(100,100,5);

cell(130,130,5);

cell(160,160,5);

cell(190,190,5);

cell(220,220,5);

cell(260,240,5);


cell(170,105,5);

cell(215,145,5);

cell(100,185,5);

cell(110,215,5);

cell(120,255,5);

for(int i=0;i<7;i++)

{

  cell(100+(i*35),0,5);

}
```

```
}




void draw_hexagon_grid(float x,float y,int ni,int nj){
  int i,j; float x0,shiftP=2.0;
  x-=((float)(ni-1))*hexagon_gx*0.5; // just shift x,y to start position (i=0,j=0)
  x-=((float)(nj-1))*hexagon_dx*0.5;
  y-=((float)(nj-1))*hexagon_gy*0.5;
  //x1=x+15*hexagon_gx*0.5,y1=y+9*hexagon_gx*0.5;
  for (x0=x,j=0;j<nj;j++,x0+=hexagon_dx+shiftP,x=x0+shiftP,y+=hexagon_gy+shiftP)
  for (i=0;i<ni;i++,x+=hexagon_gx+shiftP){
    draw_hexagon(x,y);
  }

   /* x-=((float)(ni-1))*hexagon_gx*0.5; // just shift x,y to start position (i=0,j=0)
  x-=((float)(nj-1))*hexagon_dx*0.5;
  y-=((float)(nj-1))*hexagon_gy*0.5;*/

}

void display(){
```

```
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(0.8, 0.0, 0.0);
//setBackgroundColor();
switch (f) {
  case 0:frame0();break;
  case 1:   frame1();break;
  break;
  case 2: if(c==0) {op=0;c++;}
          frame2();break;
  case 3:
        if(b==0)
        {
          yLocation = 0.0f;
          bloby=0,blobx=0,cx1=0,cy1=0,cx2=0,cy2=0;
          c0=0,c1=0,c2=0,c3=0;
          b=1;
        }
        frame3();
        break;
  case 4:
        if(b==0)
        {
          yLocation = 0.0f;
          bloby=0,blobx=0,cx1=0,cy1=0,cx2=0,cy2=0;
```

```
            c0=0,c1=0,c2=0,c3=0;
            b=1;
          }
          frame3();
          break;
    case 5:
          if(a==0){
            frameNumber=0;
            a=1;
          }
          frame4();break;
            //default:printf("boo"); /* value */:
  }
  //glFlush();
  glutSwapBuffers();
}




void frame1(){
  draw_hexagon_grid(70,70,15,13);
         if(d==1)
          drawCancerCells(70,70,12,19);
}
```

```
void drawCircle(float cx, float cy, float r, int num_segments){
  glBegin(GL_LINE_LOOP);
  for(int ii = 0; ii < num_segments; ii++)
  {
    float theta = 2.0f * 3.1415926f * (float)ii / (float)(num_segments);//get the current angle

    float x = r * cosf(theta);//calculate the x component
    float y = r * sinf(theta);//calculate the y component

    glVertex2f(x + cx, y + cy);//output vertex

  }
  glEnd();
}

void hexagon(int x,int y){
  float rad;
  float hexagon_r=20;
  float hexagon_dx=hexagon_r*cos(30.0*M_PI/180.0);
  float hexagon_dy=hexagon_r*sin(30.0*M_PI/180.0);
  float hexagon_gx=2.0*hexagon_dx;
  float hexagon_gy=2.0*hexagon_dx*sin(60.0*M_PI/180.0);

  glColor3f(0,0.8,0);
```

```
 glBegin(GL_POLYGON);
 glVertex2f(x-hexagon_dx,y-hexagon_dy);
 glVertex2f(x-hexagon_dx,y+hexagon_dy);
 glVertex2f(x          ,y+hexagon_r );
 glVertex2f(x+hexagon_dx,y+hexagon_dy);
 glVertex2f(x+hexagon_dx,y-hexagon_dy);
 glVertex2f(x          ,y-hexagon_r );
 glEnd();
}
void output(int x, int y,const char *string){
 //char string[100]="GLUT_BITMAP_TIMES_ROMAN_10";
 int len, i;
 glRasterPos2f(x, y);
 len = (int)strlen(string);
 for (i = 0; i<len; i++)
 {
   glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
 }

}

void verticleLine(int x,int y1,int y2){
 glBegin(GL_LINES);
 glVertex2f(x,y1);
 glVertex2f(x,y2);
 glEnd();
}


void hexagonCancer(int x,int y){
 float rad;
 float hexagon_r=15;
 float hexagon_dx=hexagon_r*cos(30.0*M_PI/180.0)-5;
 float hexagon_dy=hexagon_r*sin(30.0*M_PI/180.0)+2;
 float hexagon_gx=2.0*hexagon_dx;
 float hexagon_gy=2.0*hexagon_dx*sin(60.0*M_PI/180.0)+10;


 glColor3f(1,0.5,0);
 glBegin(GL_POLYGON);
 glVertex2f(x-hexagon_dx,y-hexagon_dy);
 glVertex2f(x-hexagon_dx,y+hexagon_dy);
 glVertex2f(x          ,y+hexagon_r );
 glVertex2f(x+hexagon_dx,y+hexagon_dy);
 glVertex2f(x+hexagon_dx,y-hexagon_dy);
 glVertex2f(x          ,y-hexagon_r );
 glEnd();
}
```

```
void drawCircle(GLfloat x, GLfloat y, GLfloat radius,int r,int g,int b){
  int i;
  int lineAmount = 100; //# of triangles used to draw circle

  //GLfloat radius = 0.8f; //radius
  GLfloat twicePi = 2.0f * 3.14;
  glColor3f(r,g,b);
  glBegin(GL_POLYGON);
  for(i = 0; i <= 300;i++) {
    glVertex2f(
      x + (radius * cos(i *  twicePi / lineAmount)),
      y + (radius* sin(i * twicePi / lineAmount))
    );
  }
  glEnd();
}

void drawCircleCancer(GLfloat x, GLfloat y, GLfloat radius,int r,int g,int b){
  int i;
  int lineAmount = 100; //# of triangles used to draw circle

  //GLfloat radius = 0.8f; //radius
  GLfloat twicePi = 2.0f * 3.14;
  glColor4f(r,g,b,op);
  glBegin(GL_POLYGON);
  for(i = 0; i <= 300;i++) {
    glVertex2f(
      x + (radius * cos(i *  twicePi / lineAmount)),
      y + (radius* sin(i * twicePi / lineAmount))
    );
  }
  glEnd();
}


void CellCancer(int x,int y){
  hexagonCancer(x,y);
  drawCircle(x,y,2,1,0,0);
}
void cell(int x,int y){
  hexagon(x,y);
  drawCircle(x,y,5,1,0,0);
  //drawCircle(x+10,y+5,10,1,0.5,0);
}
void cancerCellM(int x,int y){
  drawCircleCancer(20+x,10+y,13,1,1,0);
  drawCircleCancer(35+x,10+y,13,1,1,0);
  drawCircleCancer(30+x,20+y,10,1,1,0);
  drawCircleCancer(20+x,25+y,13,1,1,0);
  drawCircleCancer(15+x,10+y,10,1,1,0);

  drawCircleCancer(23+x,14+y,5,1,0,0);

}

void cancerCellB(int x,int y){

  drawCircle(20+x,10+y,13,1,1,0);

  drawCircle(35+x,10+y,13,1,1,0);

  drawCircle(30+x,20+y,10,1,1,0);

  drawCircle(20+x,25+y,13,1,1,0);

  drawCircle(15+x,10+y,10,1,1,0);

  drawCircle(23+x,14+y,5,1,0,0);

}

void melignant(){


  glColor3f(0,1,0);

  glBegin(GL_LINES);

  glVertex2f(-267,-20); // center blood vessel

  glVertex2f(-267,331); //Benign vertical


  glVertex2f(-130,330); //benign upper line

  glVertex2f(-267,330);


  glVertex2f(-267,-20);//benign lower line

  glVertex2f(-129,-20);
```

```
glVertex2f(82,329);//melignant vertical line

glVertex2f(82,80);



glVertex2f(82,330);//melignant upper line

glVertex2f(221,330);



glVertex2f(82,80);//melignant lower line

glVertex2f(221,80);



glEnd();

cancerCellM(230,100);

cancerCellM(210,140);

cancerCellM(174,170);

cancerCellM(170,215);

cancerCellM(195,255);

cancerCellM(230,280);

cancerCellM(270,270);

cancerCellM(290,230);

cancerCellM(275,120);

cancerCellM(320,180);

cancerCellM(290,150);
```

```
}
void secondSet(){
  for(i=0;i<5;i++)
  {
    hexagon(100+(i*30),100);
    drawCircle(100+(i*30),100,5,1,0,0);
  }

  for( i=0;i<4;i++)
  {
    hexagon(100+(i*30),140);
    drawCircle(100+(i*30),140,5,1,0,0);
  }

  for(int i=0;i<3;i++)
  {
    hexagon(100+(i*30),180);
    drawCircle(100+(i*30),180,5,1,0,0);
  }

  for(i=0;i<2;i++)
  {
```

```
  hexagon(100+(i*30),220);

  drawCircle(100+(i*30),220,5,1,0,0);

}


for( i=0;i<3;i++)

{

  hexagon(100+(i*30),260);

  drawCircle(100+(i*30),260,5,1,0,0);

}


for( i=0;i<5;i++)

{

  hexagon(100+(i*30),310);

  drawCircle(100+(i*30),310,5,1,0,0);

}


melignant();




glPushMatrix();


CellCancer(230,200);

CellCancer(250,200);

CellCancer(270,250);
```

```
glPopMatrix();


glPushMatrix();
CellCancer(280,220);
CellCancer(270,180);
CellCancer(290,250);
glPopMatrix();


glPushMatrix();
CellCancer(230,250);
CellCancer(290,170);
CellCancer(290,200);
glPopMatrix();


glPushMatrix();
CellCancer(250,240);
CellCancer(240,230);
CellCancer(270,150);
glPopMatrix();
//cell(265,250);
// hexagon(230,100);
// hexagon(210,140);
// hexagon(174,170);
// hexagon(170,215);
```

```
 // hexagon(195,255);

 // hexagon(230,280);

 // hexagon(270,270);

 // hexagon(290,230);

 // hexagon(275,120);

 // hexagon(300,160);


}


void firstSet(){

 for( i=0;i<5;i++)

 {

   hexagon(-250+(i*30),310);

   drawCircle(-250+(i*30),310,5,1,0,0);

 }


 for( i=0;i<4;i++)

 {

   hexagon(-250+(i*30),260);

   drawCircle(-250+(i*30),260,5,1,0,0);

 }


 for(i=0;i<3;i++)

 {
   hexagon(-250+(i*30),220);

   drawCircle(-250+(i*30),220,5,1,0,0);

 }


 for( i=0;i<2;i++)

 {

   hexagon(-250+(i*30),170);

   drawCircle(-250+(i*30),170,5,1,0,0);

 }


 for( i=0;i<2;i++)

 {

   hexagon(-250+(i*30),130);

   drawCircle(-250+(i*30),130,5,1,0,0);

 }


 for(i=0;i<2;i++)

 {

   hexagon(-250+(i*30),90);

   drawCircle(-250+(i*30),90,5,1,0,0);

 }

 for( i=0;i<4;i++)

 {

   hexagon(-250+(i*30),40);
   drawCircle(-250+(i*30),40,5,1,0,0);
 }


 for( i=0;i<5;i++)
 {
   hexagon(-250+(i*30),0);
   drawCircle(-250+(i*30),0,5,1,0,0);
 }
 cell(-90,10);
 cell(-60,40);
 cell(-45,80);
 cell(-85,290);
 cell(-65,250);
 cell(-45,220);


 cancerCellB(-110,130);
 cancerCellB(-140,70);
 cancerCellB(-190,100);
 cancerCellB(-160,150);
 cancerCellB(-100,160);
 cancerCellB(-180,110);
 cancerCellB(-200,160);
 cancerCellB(-180,130);
 cancerCellB(-150,110);
```

```
  cancerCellB(-145,190);

  cancerCellB(-170,170);

}


void drawHollowElipse(GLfloat x, GLfloat y, GLfloat radiusx,GLfloat radiusy){

  int i;

  int lineAmount = 100; //# of triangles used to draw circle


  //GLfloat radius = 0.8f; //radius

  GLfloat twicePi = 2.0f * M_PI;


  glBegin(GL_LINE_LOOP);

  for(i = 0; i <= lineAmount;i++) {

    glVertex2f(

      x + (radiusx * cos(i *  twicePi / lineAmount)),

      y + (radiusy* sin(i * twicePi / lineAmount))

    );

  }

  glEnd();

}

void drawFilledElipse(GLfloat x, GLfloat y, GLfloat radiusx,GLfloat radiusy){

  int i;

  int triangleAmount = 20; //# of triangles used to draw circle
  //GLfloat radius = 0.8f; //radius
  GLfloat twicePi = 2.0f * M_PI;


  glBegin(GL_TRIANGLE_FAN);
  glVertex2f(x, y); // center of circle
  for(i = 0; i <= triangleAmount;i++) {
    glVertex2f(
      x + (radiusx * cos(i *  twicePi / triangleAmount)),
      y + (radiusy * sin(i * twicePi / triangleAmount))
    );
  }
  glEnd();
}


void drawCig(){
  int xLL=20,yLL=40,xLR,yLR;
  /*Point p1={85,230},
  p2={115,230},
  p3={135,350},
  p4={65,350},*/
  double factor=8.0f/9.0f;
  Point p1={-5,-5},
  p2={5,-5},
  p3={5,15},
  p4={-5,15},


  p5={(p3.x+p2.x)/2.0f,(p3.y+p2.y)*factor},

  p6={(p1.x+p4.x)/2.0f,(p1.y+p4.y)*factor};

  double radiusy=1.1;

  //filled part

  glColor3f(.894f, .925f,.89f );

  glBegin(GL_POLYGON);

  glColor3f( .953, .933,.824);

  glVertex2f(p5.x,p5.y);

  glVertex2f(p6.x,p6.y);

  glVertex2f(p3.x,p3.y);

  glVertex2f(p4.x,p4.y);

  glEnd();


  //  drawFilledElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, 4);

  drawFilledElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);

  glColor3f(0,0,0);

  //  drawHollowElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,8);

  glEnd();


  // outlline of the quadrilateral

  glLineWidth(2.0f);

  glBegin(GL_LINE_LOOP);
```

```
glColor3f( 0,0,0);

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glVertex2f(p3.x,p3.y);

glVertex2f(p4.x,p4.y);

glEnd();


//lower outline

drawHollowElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, radiusy);

glLineWidth(1.0f);


//filled polygon

glBegin(GL_POLYGON);

glColor3f( .953, .933,.824);

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glVertex2f(p3.x,p3.y);

glVertex2f(p4.x,p4.y);

glEnd();


//lower  ellipse

drawFilledElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, radiusy);

drawFilledElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);

glColor3f(0,0,0);
```

```
  drawHollowElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);


  //filled area

  glBegin(GL_POLYGON);

  glColor3f(0.855f,0.651f,0.322f);

  glVertex2f(p4.x,p4.y);

  glVertex2f(p3.x,p3.y);

  glVertex2f(p5.x,p5.y);

  glVertex2f(p6.x,p6.y);

  glEnd();


  //upper filled elipse

  drawFilledElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);

  drawHollowElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);

  drawFilledElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);

  glColor3f(0,0,0);

  drawHollowElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);



}


void drawCigs(){


  //  glTranslated(-5,0,0);
  for (int i = 0; i < 4; i++) {
    glTranslated(i+12, 0, 0);
    drawCig();
  }
}
void drawFan() {
  int i,frameNumber =0;

  glRotated(frameNumber * (180.0/46), 0, 0, 1);
  glColor3f(.349f, .267f, .231f );
  for (i = 0; i < 3; i++) {
    glRotated(120, 0, 0, 1);  // Note: These rotations accumulate.
    glBegin(GL_POLYGON);
    glVertex2f(0,0);
    glVertex2f(0.5f, 0.4f);
    //glVertex2f(1.5f,0);

    glVertex2f(0.5f, -0.4f);
    /*(glVertex2f(0.3,0.2);
    //   glVertex2f(0.5f, 0.1f);
    glVertex2f(0.7f,0.2);
    glVertex2f(0.5f, -0.1f);*/
    glEnd();
    drawFilledElipse(.5,0,.15,.4);
  }
  double r=.1;
  glColor3f(     .725f, .933f, .871f);
  drawFilledCircle(0,0,(r+.05));
  glColor3f(.349f, .267f, .231f );
  drawFilledCircle(0,0,r);
}

void drawGlass(){
  int xLL=20,yLL=40,xLR,yLR;
  /*Point p1={85,230},
  p2={115,230},
  p3={135,350},
  p4={65,350},*/
  Point p1={-5,-5},
  p2={5,-5},
  p3={5,15},
  p4={-5,15},

  p5={(p3.x+p2.x)/2.0f,(p3.y+p2.y)/2.0f+1.6},
  p6={(p1.x+p4.x)/2.0f,(p1.y+p4.y)/2.0f+1.6};
  double radiusy=1.5;
  //filled part
  glColor3f(.894f, .925f,.89f );
```

```
glBegin(GL_POLYGON);

glColor3f( .953, .933,.824);

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glVertex2f(p5.x,p5.y);

glVertex2f(p6.x,p6.y);

glEnd();


//  drawFilledElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, 4);

drawFilledElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);

glColor3f(0,0,0);

//  drawHollowElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,8);

glEnd();



glLineWidth(2.0f);

glBegin(GL_LINE_LOOP);

glColor3f( 0,0,0);

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glVertex2f(p3.x,p3.y);

glVertex2f(p4.x,p4.y);

glEnd();
```

```
drawHollowElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, radiusy);

glLineWidth(1.0f);



glBegin(GL_POLYGON);

glColor3f( .953, .933,.824);

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glVertex2f(p3.x,p3.y);

glVertex2f(p4.x,p4.y);

glEnd();



//drawFilledElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, 4);

drawFilledElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);

glColor3f(0,0,0);

drawHollowElipse((p3.x+p4.x)/2,p3.y,(p4.x-p3.x)/2,radiusy);



//filled area

glBegin(GL_POLYGON);

glColor3f(0.757f ,0.471f,0.071f);

//glColor3f(.89f,.92f,.89f );

glVertex2f(p1.x,p1.y);

glVertex2f(p2.x,p2.y);

glColor3f(.965f,0.733f,0.133f);

glVertex2f(p5.x,p5.y);
```

```
 glVertex2f(p6.x,p6.y);
 glEnd();


 drawFilledElipse((p1.x+p2.x)/2, p1.y, (p2.x-p1.x)/2, radiusy);
 drawFilledElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);


 glColor3f(0.918f,0.675f,0.118f);
 drawHollowElipse((p5.x+p6.x)/2.0f,p5.y,(p6.x-p5.x)/2.0f,radiusy);




}


void drawDisk(double radius) {
 int d;
 glBegin(GL_POLYGON);
 for (d = 0; d < 32; d++) {
   double angle = 2*M_PI/32 * d;
   glVertex2d( radius*cos(angle), radius*sin(angle));
 }
 glEnd();
}


void drawSun() {
 int i;
```

```
  //glColor3f(1,1,0);


  // glColor3f( .953, .933,.824);
  glColor3f(.98,.850,.36);
  glLineWidth(3.0f);
  for (i = 0; i < 13; i++) { // Draw 13 rays, with different rotations.
    glRotatef( 360 / 13, 0, 0, 1 ); // Note that the rotations accumulate!
    glBegin(GL_LINES);
    glVertex2f(0, 0);
    glVertex2f(75, 0);
    glEnd();
  }
  glLineWidth(1.0f);
  drawDisk(40);
  glColor3f(0,0,0);
}


void Arrow(GLdouble x1,GLdouble y1,GLdouble z1,GLdouble x2,GLdouble y2,GLdouble z2,GLdouble D){
  double x=x2-x1;
  double y=y2-y1;
  double z=z2-z1;
  double L=sqrt(x*x+y*y+z*z);
  glColor3f(.694f,.859f,.133f);          //69.4, 85.9, 13.3
  GLUquadricObj *quadObj;
```

```
glPushMatrix ();


glTranslated(x1,y1,z1);
//rotation


if(theta<=120){
  theta= (frameNumber * (180.0/46));
}
glRotated(-theta, 0, 0, 1);


if((x!=0)||(y!=0)) {
  glRotated(atan2(y,x)/RADPERDEG,0.,0.,1.);
  glRotated(atan2(sqrt(x*x+y*y),z)/RADPERDEG,0.,1.,0.);
} else if (z<0){
  glRotated(180,1.,0.,0.);
}


glTranslatef(0,0,L-4*D);


quadObj = gluNewQuadric ();
gluQuadricDrawStyle (quadObj, GLU_FILL);
gluQuadricNormals (quadObj, GLU_SMOOTH);
gluCylinder(quadObj, 2*D, 0.0, 4*D, 32, 1);
```

```
gluDeleteQuadric(quadObj);


quadObj = gluNewQuadric ();

gluQuadricDrawStyle (quadObj, GLU_FILL);

gluQuadricNormals (quadObj, GLU_SMOOTH);

gluDisk(quadObj, 0.0, 2*D, 32, 1);

gluDeleteQuadric(quadObj);


glTranslatef(0,0,-L+4*D);


quadObj = gluNewQuadric ();

gluQuadricDrawStyle (quadObj, GLU_FILL);

gluQuadricNormals (quadObj, GLU_SMOOTH);

gluCylinder(quadObj, D, D, L-4*D, 32, 1);

gluDeleteQuadric(quadObj);


quadObj = gluNewQuadric ();

gluQuadricDrawStyle (quadObj, GLU_FILL);

gluQuadricNormals (quadObj, GLU_SMOOTH);

gluDisk(quadObj, 0.0, D, 32, 1);

gluDeleteQuadric(quadObj);


glPopMatrix ();
```

```
}


void drawAxes(GLdouble length){
  glPushMatrix();
  glTranslatef(-length,0,0);
  Arrow(0,0,0, 2*length,0,0, 0.2);
  glPopMatrix();


  glPushMatrix();
  glTranslatef(0,-length,0);
  Arrow(0,0,0, 0,2*length,0, 0.2);
  glPopMatrix();


  glPushMatrix();
  glTranslatef(0,0,-length);
  Arrow(0,0,0, 0,0,2*length, 0.2);
  glPopMatrix();
}


void doFrame(int v) {
  frameNumber++;
  glutPostRedisplay();
  glutTimerFunc(180,doFrame,0);
}
```

```
int main(int argc, char** argv)
{
  glutInit(&argc,argv);
  initialize();



  glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
  glutInitWindowSize(1000,1000);
  //  glutInitWindowPosition(0,0);
  glutCreateWindow("Polygon");
  //glutTimerFunc(10, Timer, 0);
  glutDisplayFunc(display);

  glutKeyboardFunc(keyboard);
  glClearColor(1.0,1.0,1.0,1.0);
  gluOrtho2D(-300,500.0,-300,500.0);

  glutTimerFunc(500,doFrame,0);
  glutMainLoop();

  return 1;
}
```

REFERENCES

1. EdwardAngle-IntractiveComputerGraphics:ATtop-DownApproching
    i. OpenGL,PublishedbyPearsonEducation,2019

2. ComputerGraphicsusingOpenGL-F.S.Hilljr.