United International University

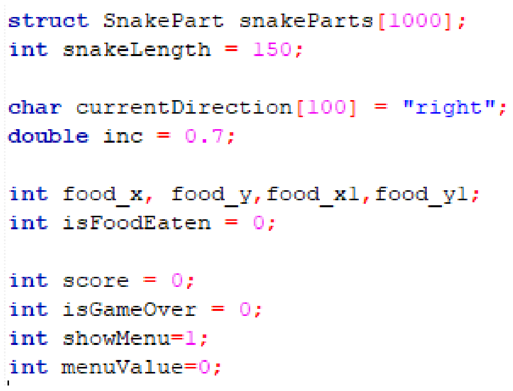
Computer Graphics Project Report
Snake Game

Submitted to: MOHAMMAD TAWHIDUL HASAN BHUIYAN

Submitted By:
Md Tofael Ahmed (011153016)
Shubhadhip Paul (011143057)
Mst Nusrat Jahan Asha (011152057)

Project Description : Our Project is a snake game. It is implemented in opengl. We have implemented two label. Now we will describe about our function.

Global Variables:

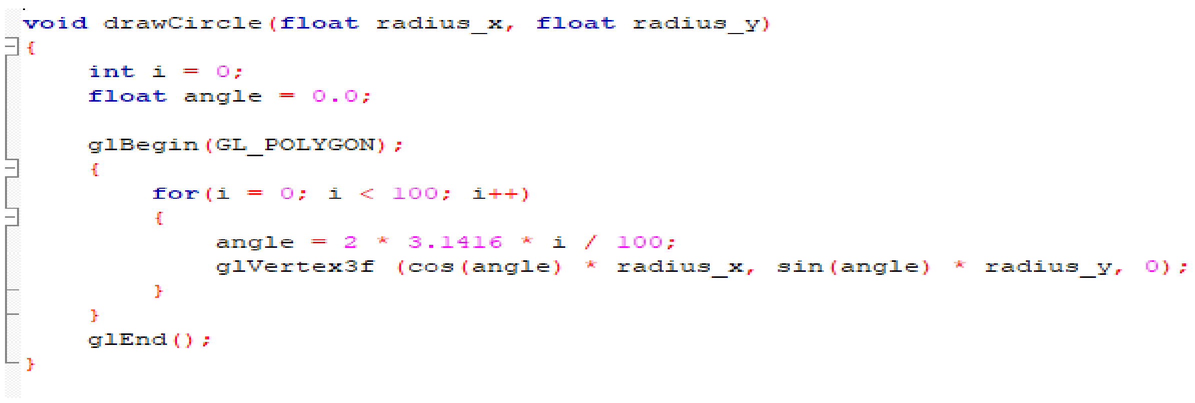


```
struct SnakePart snakeParts[1000];
int snakeLength = 150;

char currentDirection[100] = "right";
double inc = 0.7;

int food_x,  food_y,food_x1,food_y1;
int isFoodEaten = 0;

int score = 0;
int isGameOver = 0;
int showMenu=1;
int menuValue=0;
```

Here we have declared a snakeparts structure where we define snake length in x and y coordinate and the snake color. Variable snakeLength is defined to 150 where our initial snake length will be 150 pixel in x , y coordinate system.
Initial Current direction is set to right. food_x and food_y is the coordinate of where the food will place in every time randomly . The variable isFoodEaten for if the food is eaten or not then it value is set. Variable score=0 for adding score and isGameOver is set to 0 if it is become 1 then the game is over.

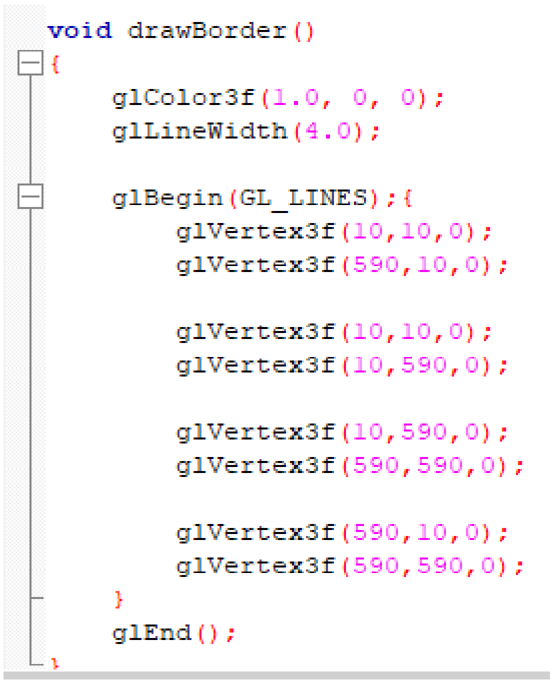
All Drawing Function:

drawCircle(float radius_x, float radius_y) : Here radius_x and radius_y is used circle radius and we have used gl_polygon function for draw square.



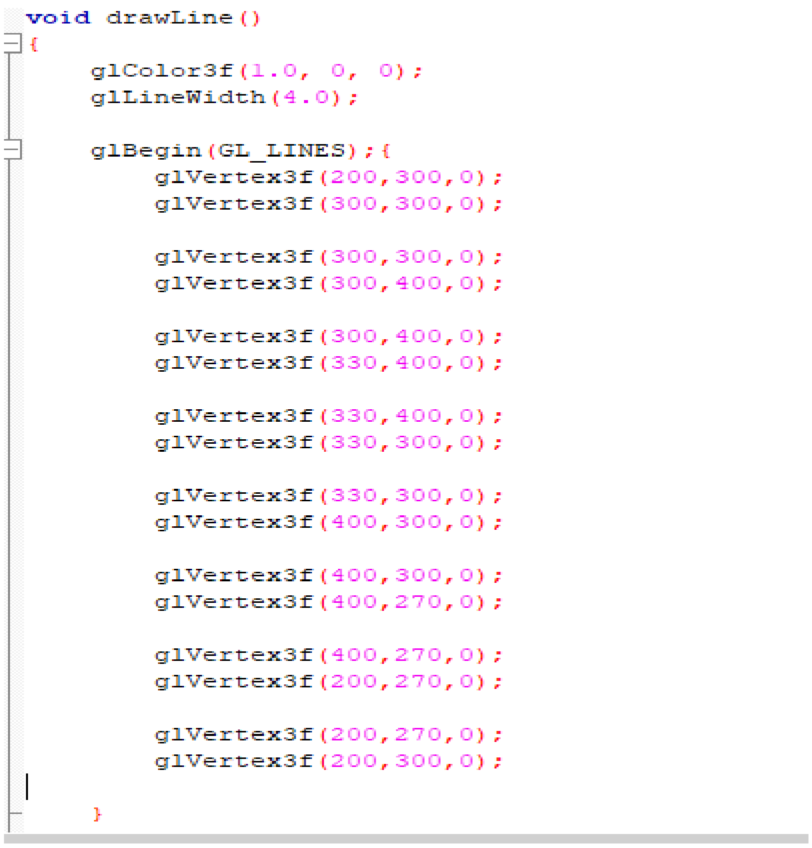
```
void drawCircle(float radius_x,  float radius_y)
{
    int i = 0;
    float angle = 0.0;

    glBegin(GL_POLYGON);
    {
        for(i = 0; i < 100; i++)
        {
            angle = 2 * 3.1416 * i / 100;
            glVertex3f (cos(angle) * radius_x, sin(angle) * radius_y, 0);
        }
    }
    glEnd();
}
```

draw*B*order() : Here we draw a rectangle border with *G*L_LINES as our default screen size is 600*600. We started from vertex point (10,10) and finish at vertex point (590,590)



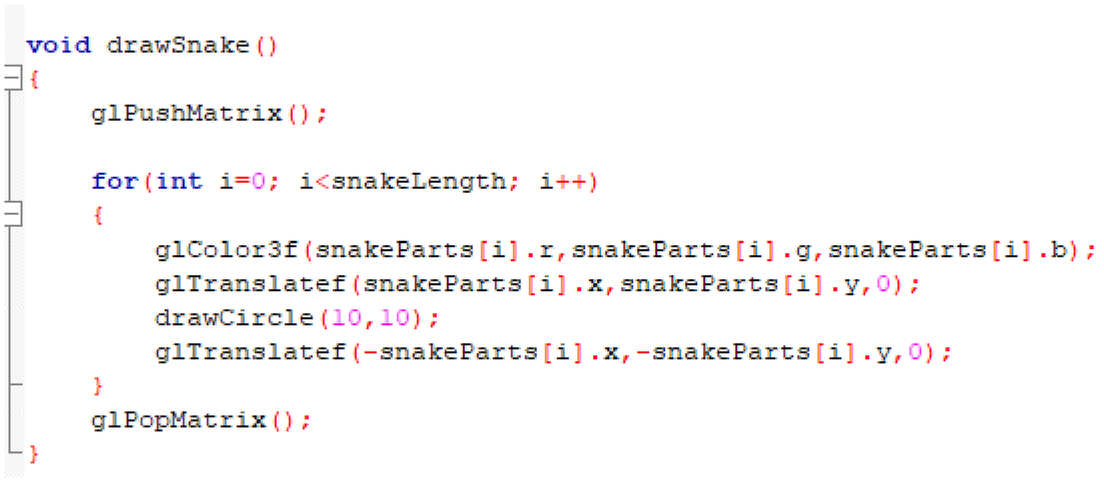
```
void drawBorder()
{
    glColor3f(1.0, 0, 0);
    glLineWidth(4.0);

    glBegin(GL_LINES);{
        glVertex3f(10,10,0);
        glVertex3f(590,10,0);

        glVertex3f(10,10,0);
        glVertex3f(10,590,0);

        glVertex3f(10,590,0);
        glVertex3f(590,590,0);

        glVertex3f(590,10,0);
        glVertex3f(590,590,0);
    }
    glEnd();
}
```

drawLine(): drawLine() function is used for drawing obstacle in our game which is used in our next label. If the snake touched in the obstacle then game will end.



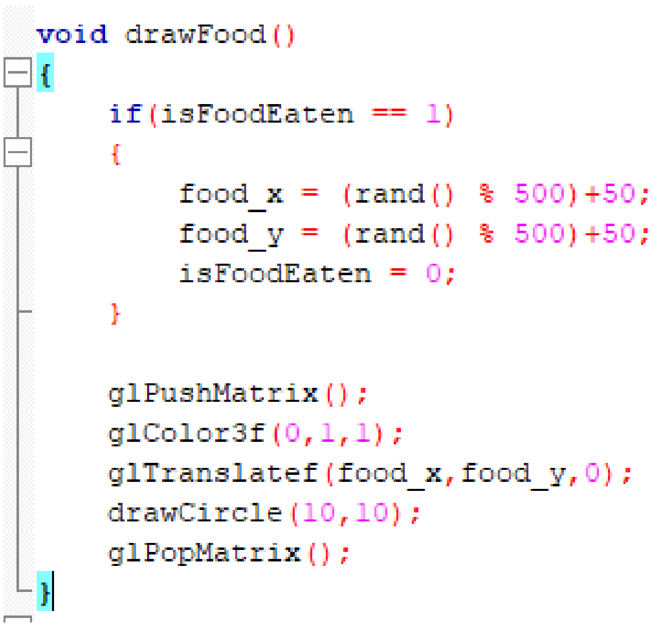
```
void drawLine()
{
    glColor3f(1.0, 0, 0);
    glLineWidth(4.0);

    glBegin(GL_LINES);{
        glVertex3f(200,300,0);
        glVertex3f(300,300,0);

        glVertex3f(300,300,0);
        glVertex3f(300,400,0);

        glVertex3f(300,400,0);
        glVertex3f(330,400,0);

        glVertex3f(330,400,0);
        glVertex3f(330,300,0);

        glVertex3f(330,300,0);
        glVertex3f(400,300,0);

        glVertex3f(400,300,0);
        glVertex3f(400,270,0);

        glVertex3f(400,270,0);
        glVertex3f(200,270,0);

        glVertex3f(200,270,0);
        glVertex3f(200,300,0);

    }
```

DrawSnake() : This function is used for drawing snake. Everytimes it will draw a snake . Here we used drawCircle() function for drawing snake in loop then it will draw till length 150 and through push and pop matrix it will draw one after another. Then we are translating every circle position.



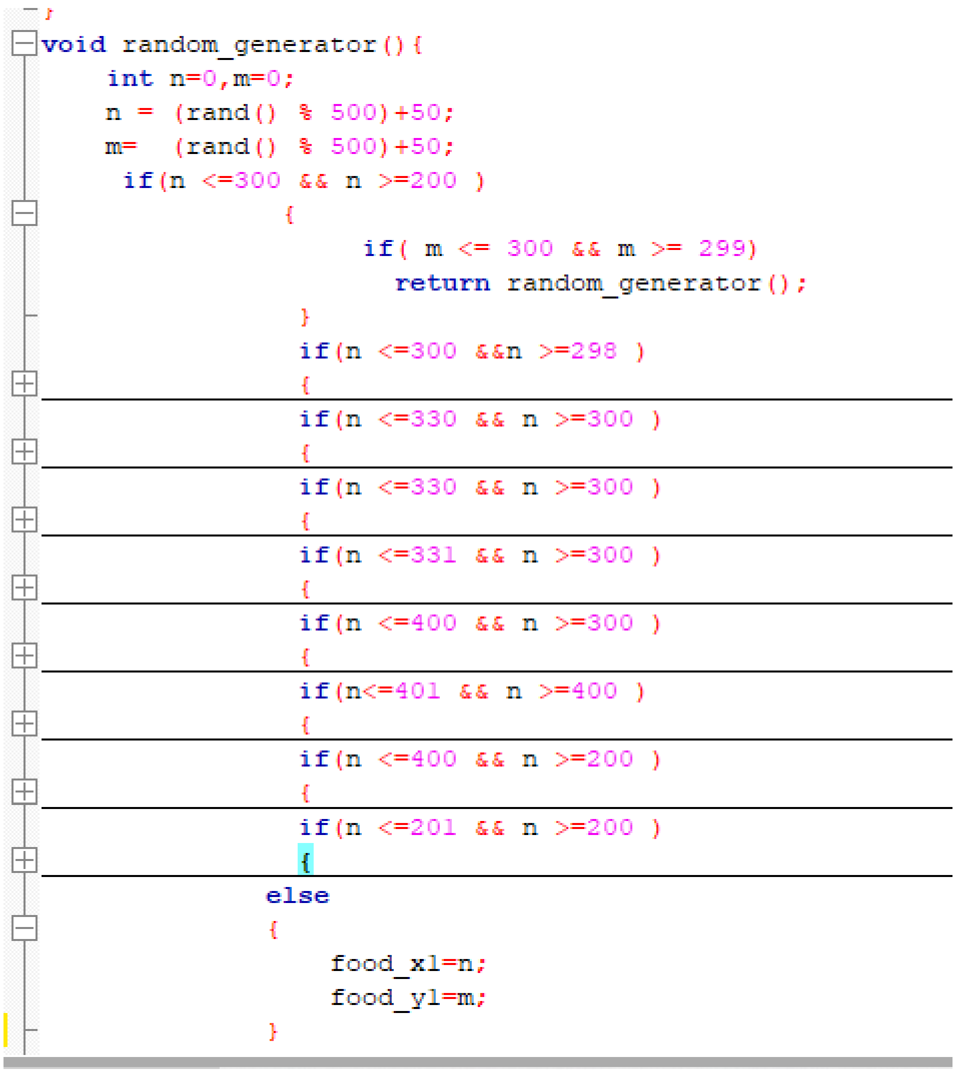
```
void drawSnake()
{
    glPushMatrix();

    for(int i=0; i<snakeLength; i++)
    {
        glColor3f(snakeParts[i].r,snakeParts[i].g,snakeParts[i].b);
        glTranslatef(snakeParts[i].x,snakeParts[i].y,0);
        drawCircle(10,10);
        glTranslatef(-snakeParts[i].x,-snakeParts[i].y,0);
    }
    glPopMatrix();
}
```

drawFood() : drawFood() is used for drawing food. Here is the food is eaten then it will draw a new food and make the food eaten value 0 for this food and food_y value is randomly generated. Then we translate the circle for drawing food as according to the randomly generated point.



```
void drawFood()
{
    if(isFoodEaten == 1)
    {
        food_x = (rand() % 500)+50;
        food_y = (rand() % 500)+50;
        isFoodEaten = 0;
    }

    glPushMatrix();
    glColor3f(0,1,1);
    glTranslatef(food_x,food_y,0);
    drawCircle(10,10);
    glPopMatrix();
}
```

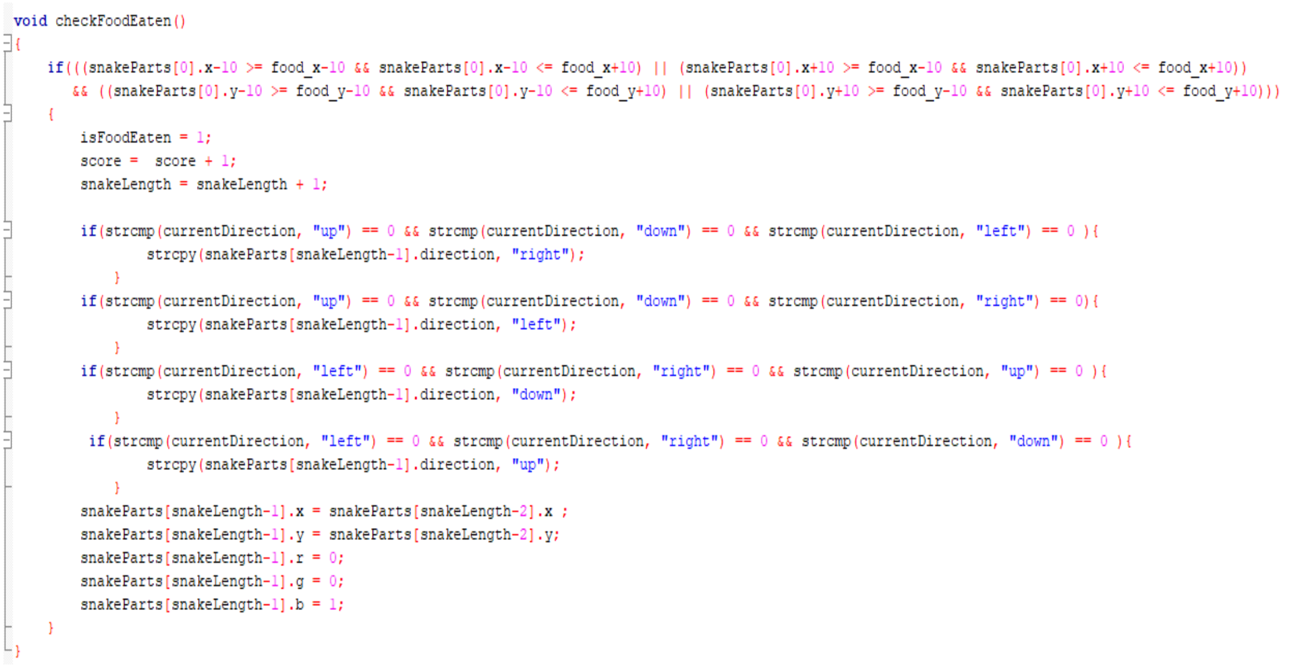
random_generator(): This function is used for random generated number for drawing food in level 2 as there is an obstacle in our level 2 . So if the food is draw inside this obstacle, Snake can’t eat it. That’s why it will coordinate outside the obstacle. We done this as recursive function.



```
}
void random_generator(){
    int n=0,m=0;
    n = (rand() % 500)+50;
    m=  (rand() % 500)+50;
     if(n <=300 && n >=200 )
                {
                        if( m <= 300 && m >= 299)
                          return random_generator();
                 }
                 if(n <=300 &&n >=298 )
                 {
                 if(n <=330 && n >=300 )
                 {
                 if(n <=330 && n >=300 )
                 {
                 if(n <=331 && n >=300 )
                 {
                 if(n <=400 && n >=300 )
                 {
                 if(n<=401 && n >=400 )
                 {
                 if(n <=400 && n >=200 )
                 {
                 if(n <=201 && n >=200 )
                 {
              else
              {
                  food_x1=n;
                  food_y1=m;
              }
```

draw_Food1() : It is same as draw food function but we use it in level 2 because for the random generated number are being generated from another function for food_x and food_y as it can't be drawn inside the obstacle.

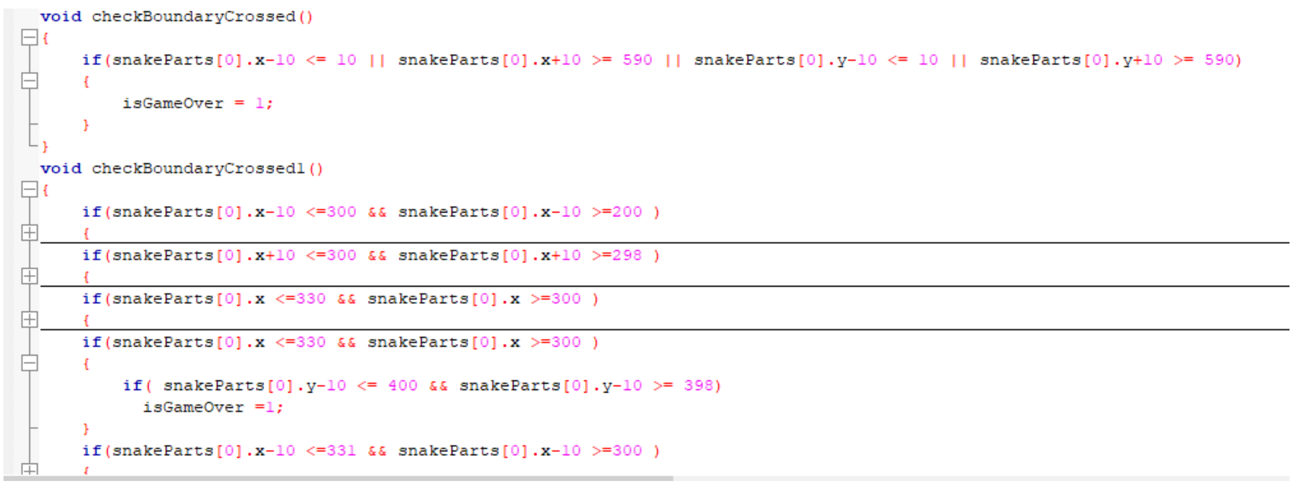
CheckFoodEaten() : this is function is used for checking the food is eaten by snake if it is eaten then we make the isFoodEaten to 1 then increase the score and snake length. Then we compare the current snake direction to left, right, up or down then we set the snake direction of the last which is increased by for food eaten to snake current position. Then we add the last drawsnake to snake current length. And we set the snake last part colors to blue for increasing part.



```
void checkFoodEaten()
{
    if(((snakeParts[0].x-10 >= food_x-10 && snakeParts[0].x-10 <= food_x+10) || (snakeParts[0].x+10 >= food_x-10 && snakeParts[0].x+10 <= food_x+10))
       && ((snakeParts[0].y-10 >= food_y-10 && snakeParts[0].y-10 <= food_y+10) || (snakeParts[0].y+10 >= food_y-10 && snakeParts[0].y+10 <= food_y+10)))
    {
        isFoodEaten = 1;
        score =  score + 1;
        snakeLength = snakeLength + 1;

        if(strcmp(currentDirection, "up") == 0 && strcmp(currentDirection, "down") == 0 && strcmp(currentDirection, "left") == 0 ){
                strcpy(snakeParts[snakeLength-1].direction, "right");
            }
        if(strcmp(currentDirection, "up") == 0 && strcmp(currentDirection, "down") == 0 && strcmp(currentDirection, "right") == 0){
                strcpy(snakeParts[snakeLength-1].direction, "left");
            }
        if(strcmp(currentDirection, "left") == 0 && strcmp(currentDirection, "right") == 0 && strcmp(currentDirection, "up") == 0 ){
                strcpy(snakeParts[snakeLength-1].direction, "down");
            }
         if(strcmp(currentDirection, "left") == 0 && strcmp(currentDirection, "right") == 0 && strcmp(currentDirection, "down") == 0 ){
                strcpy(snakeParts[snakeLength-1].direction, "up");
            }
        snakeParts[snakeLength-1].x = snakeParts[snakeLength-2].x ;
        snakeParts[snakeLength-1].y = snakeParts[snakeLength-2].y;
        snakeParts[snakeLength-1].r = 0;
        snakeParts[snakeLength-1].g = 0;
        snakeParts[snakeLength-1].b = 1;
    }
}
```

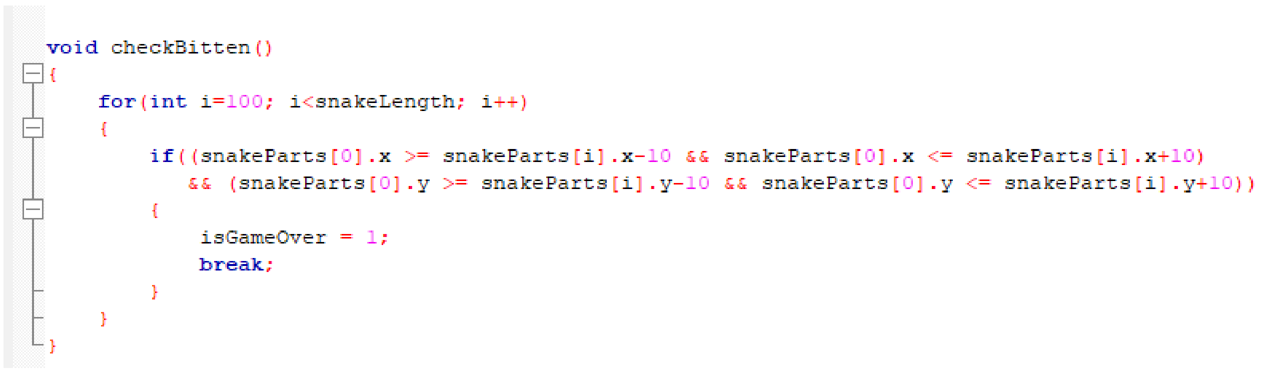
CheckBoundaryCrossed() : This function is used for if the snake initial position is outside of boundary or not if it is outside the boundary then the game is over and we are checking by adding +10 or -10 to x and Y position because of our snake is drawn by the square and which radius is 10.

checkBoundaryCrossed1() is used for checking the boundary in second label for the obstacle. As the snake touch any side of the boundary the game will over.



```
void checkBoundaryCrossed()
{
    if(snakeParts[0].x-10 <= 10 || snakeParts[0].x+10 >= 590 || snakeParts[0].y-10 <= 10 || snakeParts[0].y+10 >= 590)
    {
        isGameOver = 1;
    }
}
void checkBoundaryCrossed1()
{
    if(snakeParts[0].x-10 <=300 && snakeParts[0].x-10 >=200 )
    {
    if(snakeParts[0].x+10 <=300 && snakeParts[0].x+10 >=298 )
    {
    if(snakeParts[0].x <=330 && snakeParts[0].x >=300 )
    {
    if(snakeParts[0].x <=330 && snakeParts[0].x >=300 )
    {
        if( snakeParts[0].y-10 <= 400 && snakeParts[0].y-10 >= 398)
          isGameOver =1;
    }
    if(snakeParts[0].x-10 <=331 && snakeParts[0].x-10 >=300 )
    {
```

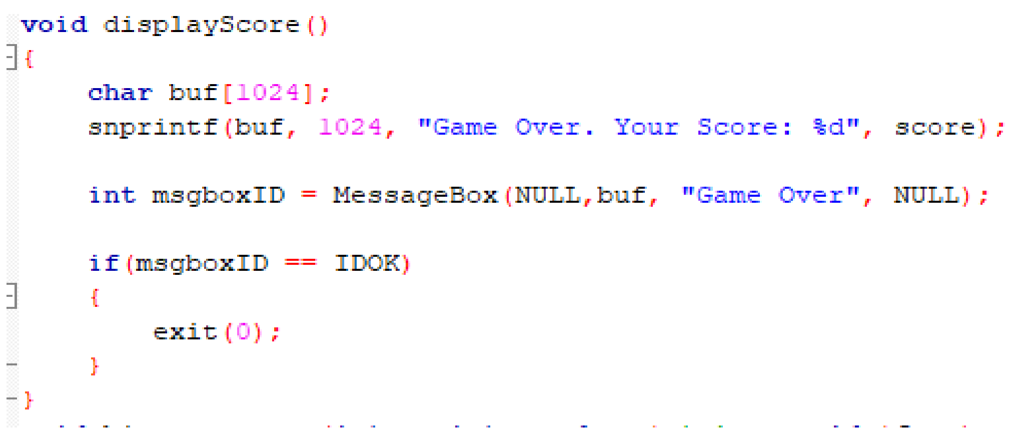
CheckBitten() : This function is used for the snake bites its own body here is define some condition which is defined that if snake head position is same as



```
void checkBitten()
{
    for(int i=100; i<snakeLength; i++)
    {
        if((snakeParts[0].x >= snakeParts[i].x-10 && snakeParts[0].x <= snakeParts[i].x+10)
           && (snakeParts[0].y >= snakeParts[i].y-10 && snakeParts[0].y <= snakeParts[i].y+10))
        {
            isGameOver = 1;
            break;
        }
    }
}
```

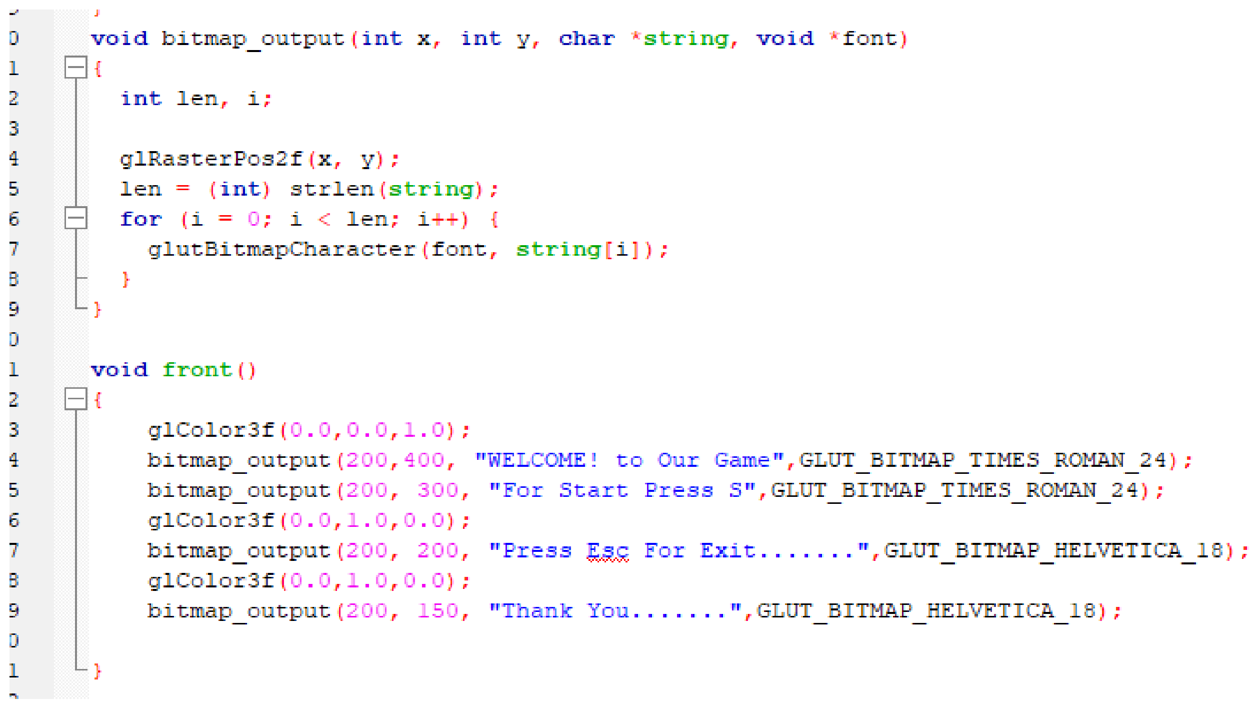
body position then the game will over.

DisplayScore(): This function is used for displaying the score after game over.



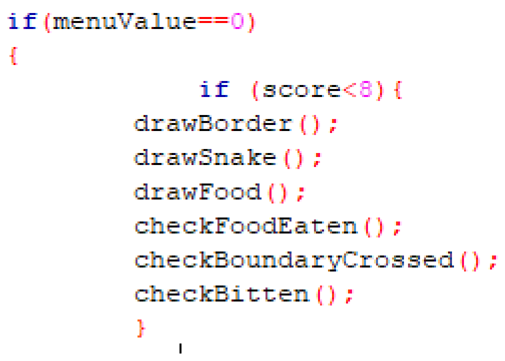
```
void displayScore()
{
    char buf[1024];
    snprintf(buf, 1024, "Game Over. Your Score: %d", score);

    int msgboxID = MessageBox(NULL,buf, "Game Over", NULL);

    if(msgboxID == IDOK)
    {
        exit(0);
    }
}
```

Void front(): It is used for displaying the initial screen for our game . For printing we have used bitmap_output function then the coordinate and the fonts for the line.



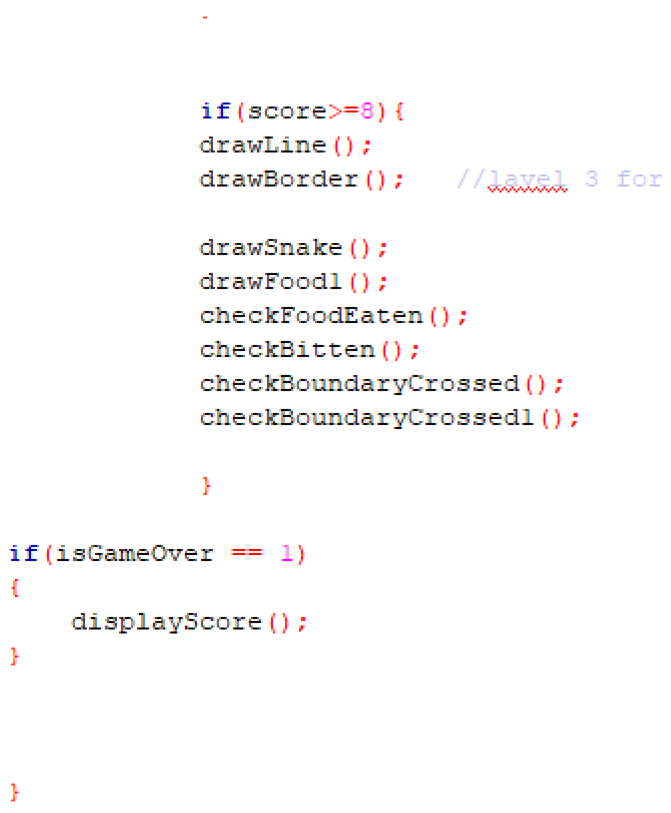
```
void bitmap_output(int x, int y, char *string, void *font)
{
  int len, i;

  glRasterPos2f(x, y);
  len = (int) strlen(string);
  for (i = 0; i < len; i++) {
    glutBitmapCharacter(font, string[i]);
  }
}

void front()
{
    glColor3f(0.0,0.0,1.0);
    bitmap_output(200,400, "WELCOME! to Our Game",GLUT_BITMAP_TIMES_ROMAN_24);
    bitmap_output(200, 300, "For Start Press S",GLUT_BITMAP_TIMES_ROMAN_24);
    glColor3f(0.0,1.0,0.0);
    bitmap_output(200, 200, "Press Esc For Exit.......",GLUT_BITMAP_HELVETICA_18);
    glColor3f(0.0,1.0,0.0);
    bitmap_output(200, 150, "Thank You.......",GLUT_BITMAP_HELVETICA_18);

}
```

Display(): In display function we have used this if the score less then 8 then it play to label 1 and then it will go to label 2. then the all function is called.



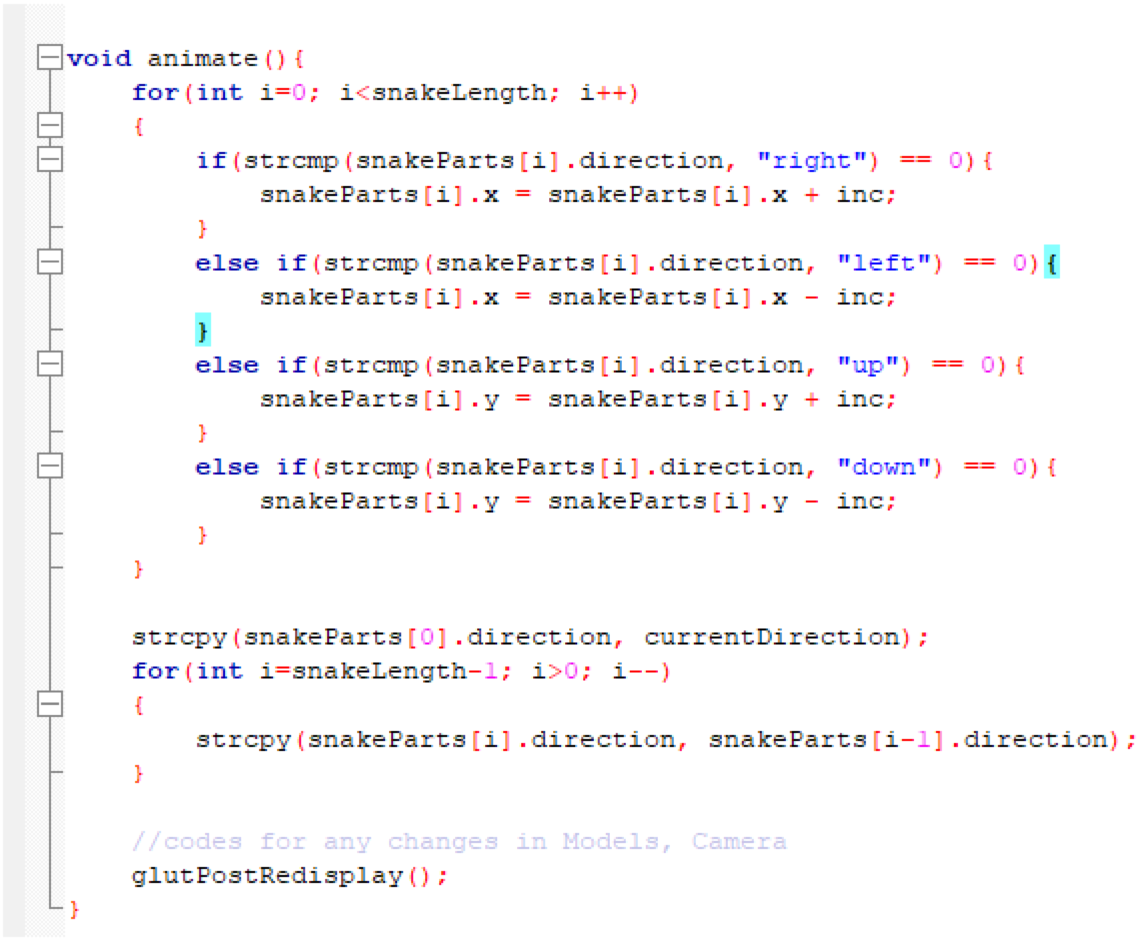
```
if(menuValue==0)
{
            if (score<8){
        drawBorder();
        drawSnake();
        drawFood();
        checkFoodEaten();
        checkBoundaryCrossed();
        checkBitten();
        }
```

Then if the score go to greater than 8 it will go to next label and here we have used here drawfood1() for not generating in food inside the obstacle. Then if the global variable isGameOver set to 1 for the snake boundary crossed or self bitten . Then the game will over and Display the score.



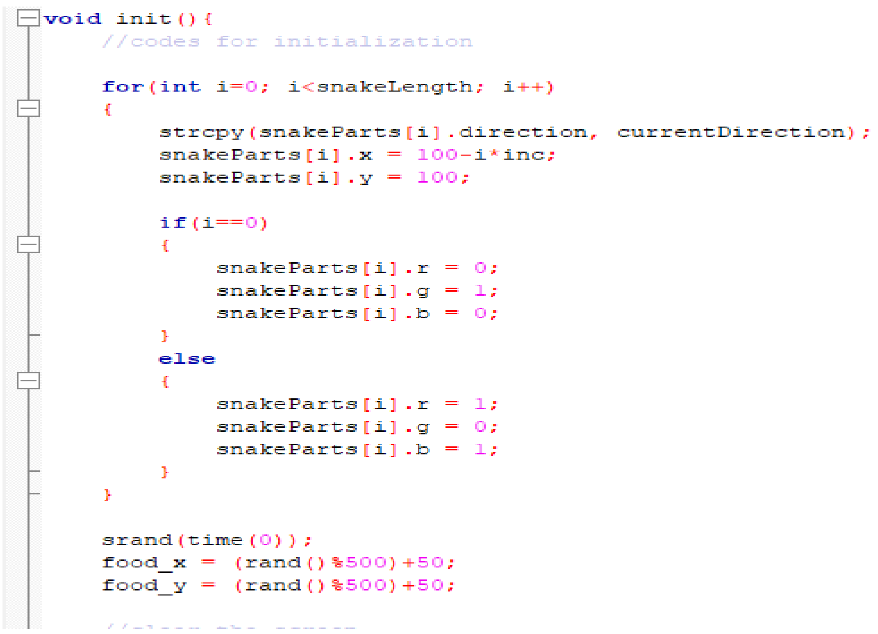
```
                        if(score>=8){
                        drawLine();
                        drawBorder();    //lavel 3 for

                        drawSnake();
                        drawFood1();
                        checkFoodEaten();
                        checkBitten();
                        checkBoundaryCrossed();
                        checkBoundaryCrossed1();

                        }

            if(isGameOver == 1)
            {
                displayScore();
            }



            }
```

void animate(): This function is used for animating the snake.



```
void animate(){
    for(int i=0; i<snakeLength; i++)
    {
        if(strcmp(snakeParts[i].direction, "right") == 0){
            snakeParts[i].x = snakeParts[i].x + inc;
        }
        else if(strcmp(snakeParts[i].direction, "left") == 0){
            snakeParts[i].x = snakeParts[i].x - inc;
        }
        else if(strcmp(snakeParts[i].direction, "up") == 0){
            snakeParts[i].y = snakeParts[i].y + inc;
        }
        else if(strcmp(snakeParts[i].direction, "down") == 0){
            snakeParts[i].y = snakeParts[i].y - inc;
        }
    }

    strcpy(snakeParts[0].direction, currentDirection);
    for(int i=snakeLength-1; i>0; i--)
    {
        strcpy(snakeParts[i].direction, snakeParts[i-1].direction);
    }

    //codes for any changes in Models, Camera
    glutPostRedisplay();
}
```

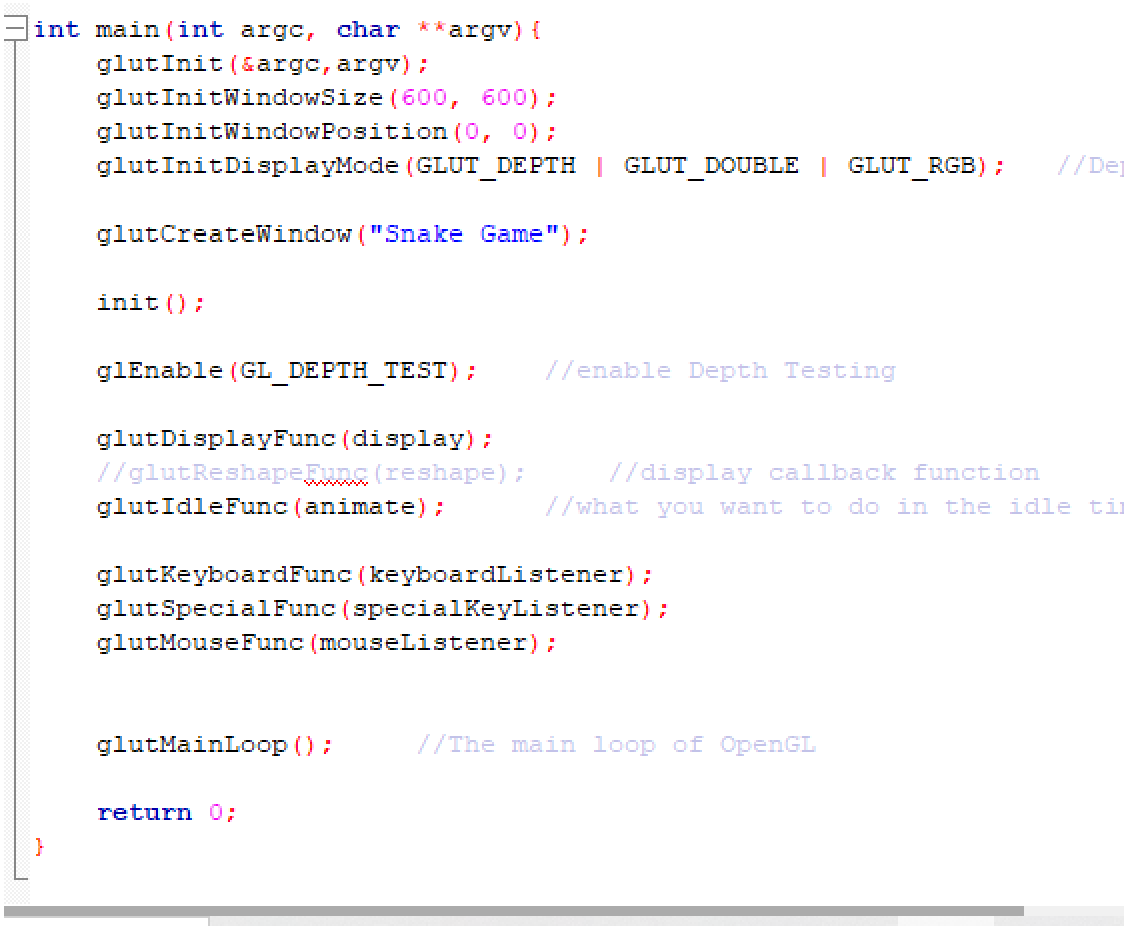
init(): this function is used for primary initialization if snakeparts[i] = then we are setting the first head color to green then we set different color for other part of the snake and then initially a food will be called and srand(time(0)) is used for it can call random number



```
void init(){
    //codes for initialization

    for(int i=0; i<snakeLength; i++)
    {
        strcpy(snakeParts[i].direction, currentDirection);
        snakeParts[i].x = 100-i*inc;
        snakeParts[i].y = 100;

        if(i==0)
        {
            snakeParts[i].r = 0;
            snakeParts[i].g = 1;
            snakeParts[i].b = 0;
        }
        else
        {
            snakeParts[i].r = 1;
            snakeParts[i].g = 0;
            snakeParts[i].b = 1;
        }
    }

    srand(time(0));
    food_x = (rand()%500)+50;
    food_y = (rand()%500)+50;
```

only once time.

This is the main function .Where game screensize to set 600*600.Then Here all the function is called



```
int main(int argc, char **argv){
    glutInit(&argc,argv);
    glutInitWindowSize(600, 600);
    glutInitWindowPosition(0, 0);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB);    //De

    glutCreateWindow("Snake Game");

    init();

    glEnable(GL_DEPTH_TEST);     //enable Depth Testing

    glutDisplayFunc(display);
    //glutReshapeFunc(reshape);      //display callback function
    glutIdleFunc(animate);       //what you want to do in the idle ti

    glutKeyboardFunc(keyboardListener);
    glutSpecialFunc(specialKeyListener);
    glutMouseFunc(mouseListener);


    glutMainLoop();      //The main loop of OpenGL

    return 0;
}
```