Figure 21: Character moving along the Circumference

TO MOVE A CHARACTER ALONG THE RADIUS

The following C program shows how to move a character along the radius.

```
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
void main()
{
  int x_cor, y_cor, radius;
  int i;
  int gdriver=DETECT, gmode, errorcode; // request auto declaration
  /* read result of initialization */
```

```
errorcode = graphresult();
if (errorcode != grOk)  /* an error occurred */
{
 printf("Graphics error: %s\n", grapherrormsg(errorcode));
 printf("Press any key to halt:");
  getch();
  exit(1); /* terminate with an error code */
}
// Input x and y co-ordinate for the circle
printf("Enter the x and y co-ordinate:");
scanf("%d %d", &x_cor, &y_cor);
// Input radius of a circle
printf("Enter radius:");
scanf("%d", &radius);
initgraph(&gdriver, &gmode, "C:\\tc\\bgi");   // initialize graphics and local variables
// Set Circle color
setcolor(BLUE);
// Draw Circle
circle(x_cor, y_cor, radius);
// Set Line color
setcolor(WHITE);
// Draw Line
line(x_cor, y_cor, x_cor+radius, y_cor);
// Moving the character along the radius
for(i=x_cor; i<x_cor+radius; i++)
{
      setcolor(RED);
      outtextxy(i, y_cor-10, "S");
      delay(150);
      setcolor(BLACK);
      outtextxy(i, y_cor-10, "S");
}
closegraph();
getch();
}
```

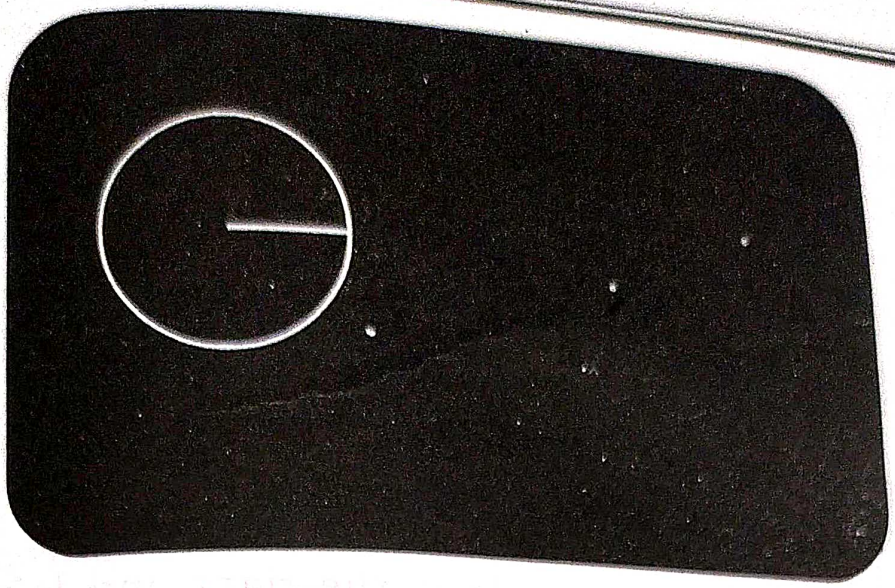

Figure 23: Character moving along the Radius

where L_{tx} and L_{ty} ... direction.

The following C program enter a line, rectangle and triangle coordinates and then translates them according to the user's enter choice.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int x1, x2, x3, x4, y1, y2, y3, y4, x_axis, y_axis;
int choice;
int main(void)
{
 int gdriver = DETECT, gmode;
 initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
 clrscr();
printf(" MAIN-MENU\n");
printf("\t TRANSLATION\n");
printf("\t 1. LINE\n");
printf("\t 2. RECTANGLE\n");
printf("\t 3. TRIANGLE\n");
printf("Enter your choice:");
scanf("%d", &choice);
switch(choice)
{
// For line
case 1:
        printf("Enter the values of line coordinates(x1, y1, x2, y2):");
```

```
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        printf("Enter the translation coordinates(x_axis, y_axis):");
        scanf("%d %d", &x_axis, &y_axis);
        cleardevice();
        line(x1,y1,x2,y2);
        // Line Translation
        printf("Now hit a key to see translation:");
        getch();
        line(x1+x_axis,y1+y_axis,x2+x_axis,y2+y_axis);
        break;
// For Rectangle
case 2:
        printf("Enter the top left coordinates(x1, y1):");
        scanf("%d %d", &x1, &y1);
        printf("Enter the bottom right coordinates(x2, y2):");
        scanf("%d %d", &x2, &y2);
        printf("Enter the values of translation coordinates(x_axis, y_axis):\n");
        scanf("%d %d", &x_axis, &y_axis);
        cleardevice();
        rectangle(x1,y1,x2,y2);
        // Rectangle Translation
        printf("Now hit a key to see translation:");
        getch();
        rectangle(x1+x_axis,y1+y_axis,x2+x_axis,y2+y_axis);
        break;
// For Triangle
case 3:
        printf("Enter coordinates of line1(x1, y1, x2, y2):\n");
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        printf("Enter coordinates for relative line(x3, y3):\n");
        scanf("%d %d", &x3, &y3);
        printf("Enter translation coordinates(x_axis, y_axis):\n");
        scanf("%d %d", &x_axis, &y_axis);
        cleardevice();
        line(x1,y1,x2,y2);
```

```
        moveto(x2,y2);
        lineto(x3,y3);
        moveto(x3,y3);
        lineto(x1,y1);
        // Triangle Translation
        printf("Now hit a key to see translation:");
        getch();
        moveto(x1+x_axis,y1+y_axis);
        lineto(x2+x_axis,y2+y_axis);
        moveto(x2+x_axis,y2+y_axis);
        lineto(x3+x_axis,y3+y_axis);
        moveto(x3+x_axis,y3+y_axis);
        lineto(x1+x_axis,y1+y_axis);
        break;
    default:
        printf("Invalid choice");
        break;
    }
    getch();
}
```

SCALING

The geometrical transformation in which the size of the picture chan i.e. increased or decreased is called SCALING.

Scaling is achieved by multiplying the co-ordinates of the picture scaling factor.

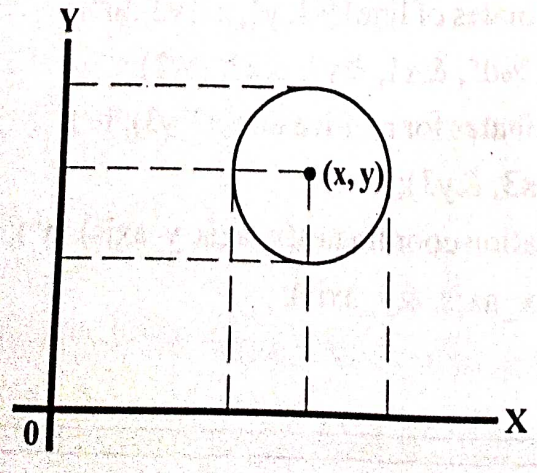

Figure 3 : Original Pictur

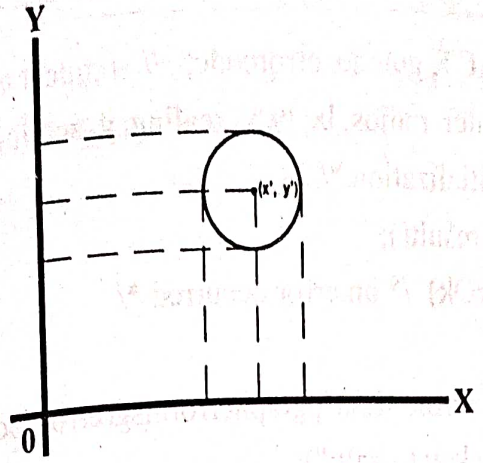

Figure 4: Picture after Scaling

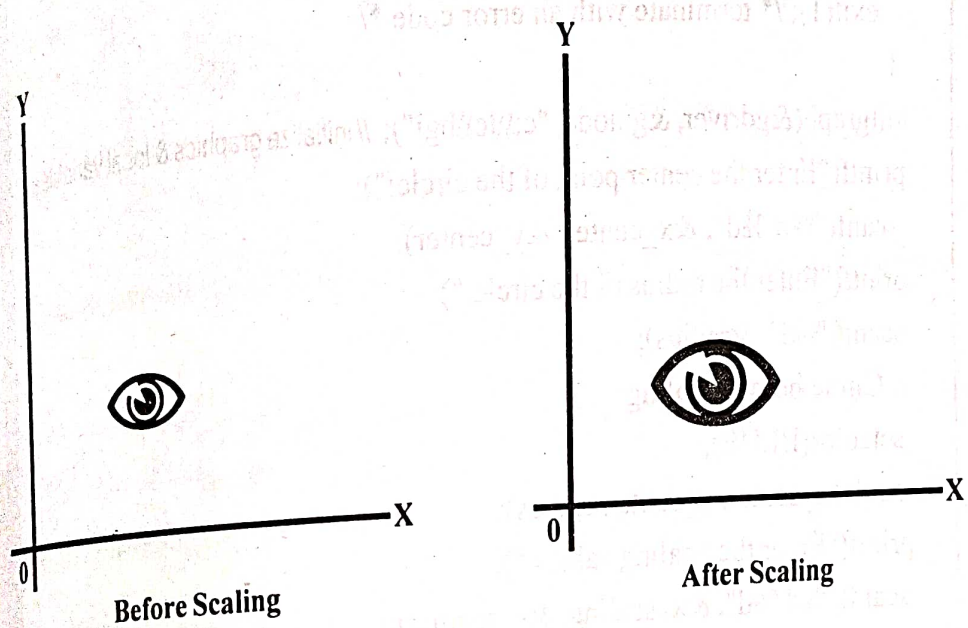

Figure 5: Pictorial view of Scaling

For a two dimensional picture we have two scaling factors S_{fx} and S_{fy}.

If the co-ordinates of original picture is x and y then that of new picture will be:

$$x' = x * S_{fx}$$

$$y' = y * S_{fy}$$

The following C program shows the scaling of an object.

```
#include<graphics.h>
#include<stdio.h>
void main()
{
```

```
int gdriver=DETECT, gmode, errorcode;  //  request auto declaration
int x_center, y_center, radius, lx, ly, x_scaling, y_scaling;
/* read result of initialization */
errorcode = graphresult();
if (errorcode != grOk)  /* an error occurred */
{
 printf("Graphics error: %s\n", grapherrormsg(errorcode));
 printf("Press any key to halt:");
  getch();
  exit(1); /* terminate with an error code */
}
initgraph(&gdriver, &gmode, "c:\\tc\\bgi");  // initialize graphics & local variables
printf("Enter the center point of the circle:");
 scanf("%d %d", &x_center, &y_center);
printf("Enter the radius of the circle:");
 scanf("%d", &radius);
 // Circle before Scaling
 setcolor(BLUE);
 circle(x_center, y_center, radius);
 printf("Enter the Scaling values:");
 scanf("%d %d", &x_scaling, &y_scaling);
 lx=radius*x_scaling;
 ly=radius*y_scaling;
  // Circle after Scaling
  setcolor(RED);
 ellipse(x_center, y_center, 0, 360, lx, ly);
  getch();
}
```

The output would be:

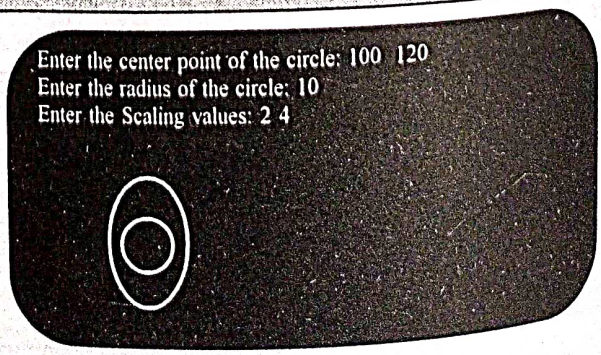

Figure 6: Screen showing Scaling of an object

The following C program reflects an object.

```
#include<graphics.h>
#include<stdio.h>
void main()
{
  int gdriver=DETECT, gmode, errorcode;  //  request auto declaration
  int x, y, ox, oy, p;
  /* read result of initialization */
  errorcode = graphresult();
  if (errorcode != grOk)  /* an error occurred */
  {
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
    getch();
   exit(1); /* terminate with an error code */
  }
```

```
initgraph(&gdriver, &gmode, "c:\\tc\\bgi");  // initialize graphics & local variables
 // Drawing lines
 setcolor(RED);
line(310, 0, 310, 480);
 setcolor(RED);
line(0, 250, 660, 250);
 // Input x and y co-ordinate for the point
printf("Enter x and y co-ordinate of the point");
 scanf("%d %d", &x, &y);
ox=310+x;
oy=250-y;
putpixel(ox, oy, WHITE);
printf("Enter the position of the mirror(0 for x axis, 1 for y axis)");
 scanf("%d", &p);
 // Reflection along the x-axis
 if(p==0)
 {
   oy=250+y;
   putpixel(ox, oy, WHITE);
 }
 // Reflection along the y-axis
 if(p==1)
 {
   ox=310-x;
   putpixel(ox, oy, WHITE);
 }
 getch();
}
```

ROTATION

The geometrical transformation in which the picture is rotated through an angle from its original position is called ROTATION.

The following object is rotated along a circular path about a fixed point called Pivot Point.

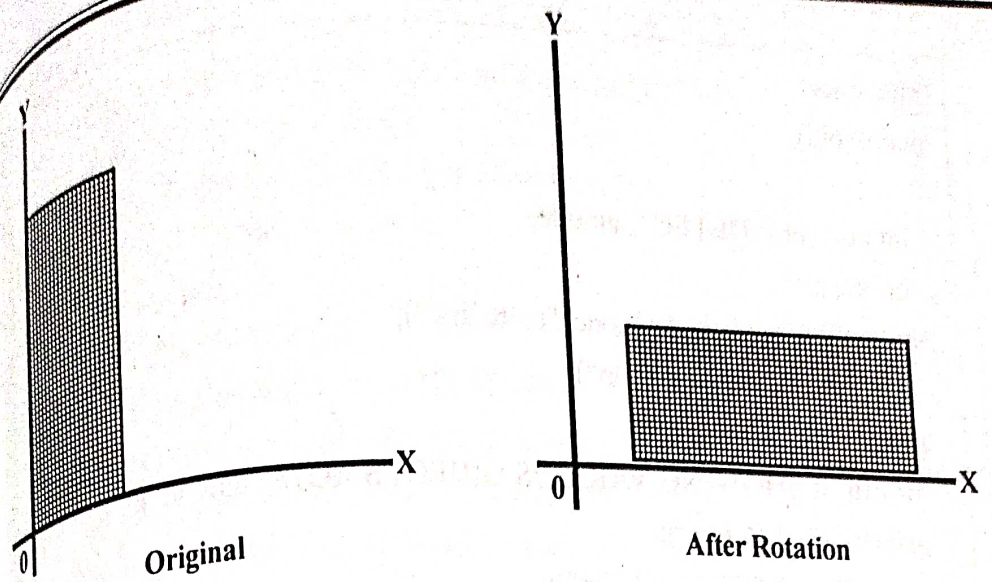

Figure 8: Rotation

The object is rotated about a pivot point (0, 0).

The angle through which the picture is rotated is called Rotation Angle.

If (x, y) are the co-ordinates of the original picture and x', y' are the co-ordinates of the picture after rotation and θ is the angle of rotation then we have the rotation equation as:

$$x' = x \cos \theta - y \sin \theta \quad \text{..... (i)}$$

$$y' = x \cos \theta + y \sin \theta \quad \text{..... (ii)}$$

Condition I

When θ is positive, rotation is in anti-clockwise direction.

Condition II

When θ is negative, rotation is in clockwise direction.

The following C program enter a line, rectangle and triangle co-ordinates and then rotates them according to the user's enter choice.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
float x1, x2, x3, x4, y1, y2, y3, y4, angle_of_rot;
```

```
int choice;
main(void)
{
 int gdriver = DETECT, gmode;
 clrscr();
 initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
 printf("\t MAIN-MENU\n");
 printf("\n");
 printf("\t SHOWING VARIOUS OBJECT'S ROTATION \n");
 printf("\t 1. LINE\n");
 printf("\t 2. RECTANGLE\n");
 printf("\t 3. TRIANGLE\n");
 scanf("%d", &choice);
 switch(choice)
 {
 case 1:
      printf("Enter the values of line coordinates(x1, y1, x2, y2):");
      scanf("%f %f %f %f", &x1, &y1, &x2, &y2);
      printf("Enter the value for angle of rotation:");
      scanf("%f", &angle_of_rot);
      cleardevice();
      line(x1,y1,x2,y2);
      angle_of_rot=angle_of_rot*(3.14/180);
      x1=(x1*cos(angle_of_rot))-(y1*sin(angle_of_rot));
      y1=(x1*sin(angle_of_rot))+(y1*cos(angle_of_rot));
      x2=(x2*cos(angle_of_rot))-(y2*sin(angle_of_rot));
      y2=(x2*sin(angle_of_rot))+(y2*cos(angle_of_rot));
      printf("Hit a key to see rotation:");
      getch();
      line(x1,y1,x2,y2);
      break;
 case 2:
      printf("Enter the top left coordinates(x1, y1):");
      scanf("%f %f", &x1, &y1);
```



```
      printf("Enter the bottom right coordinates(x2, y2):");
      scanf("%f %f", &x2, &y2);
      printf("Enter the value for angle of rotation:");
      scanf("%f", &angle_of_rot);
      cleardevice();
      rectangle(x1,y1,x2,y2);
      angle_of_rot=angle_of_rot*(3.14/180);
      x1=(x1*cos(angle_of_rot))-(y1*sin(angle_of_rot));
      y1=(x1*sin(angle_of_rot))+(y1*cos(angle_of_rot));
      x2=(x2*cos(angle_of_rot))-(y2*sin(angle_of_rot));
      y2=(x2*sin(angle_of_rot))+(y2*cos(angle_of_rot));
      printf("Hit a key to see rotation:");
      getch();
      rectangle(x1,y1,x2,y2);
      break;
 case 3:
      printf("Enter coordinates of line1(x1, y1, x2, y2):\n");
      scanf("%f %f", &x1, &y1, &x2, &y2);
      printf("Enter coordinates for relative line(x3, y3):\n");
      scanf("%f %f", &x3, &y3);
      printf("Enter the angle of rotation:\n");
      scanf("%f", &angle_of_rot);
      cleardevice();
      line(x1,y1,x2,y2);
      moveto(x2,y2);
      lineto(x3,y3);
      moveto(x3,y3);
      lineto(x1,y1);
      angle_of_rot=angle_of_rot*(3.14/180);
      x1=(x1*cos(angle_of_rot))-(y1*sin(angle_of_rot));
      y1=(x1*sin(angle_of_rot))+(y1*cos(angle_of_rot));
      x2=(x2*cos(angle_of_rot))-(y2*sin(angle_of_rot));
      y2=(x2*sin(angle_of_rot))+(y2*cos(angle_of_rot));
      x3=(x3*cos(angle_of_rot))-(y3*sin(angle_of_rot));
```

```
        y3=(x3*sin(angle_of_rot))+(y3*cos(angle_of_rot));
        printf("Hit a key to see rotation:");
        getch();
        moveto(x1,y1);
        lineto(x2,y2);
        moveto(x2,y2);
        lineto(x3,y3);
        moveto(x3,y3);
        lineto(x1,y1);
        break;
default:
        printf("Invalid choice");
        break;
}
getch();
}
```

SHEAR

An additional operation called Shear is also useful for forming general transformations. Figure 9 shows the shearing along the x-axis.

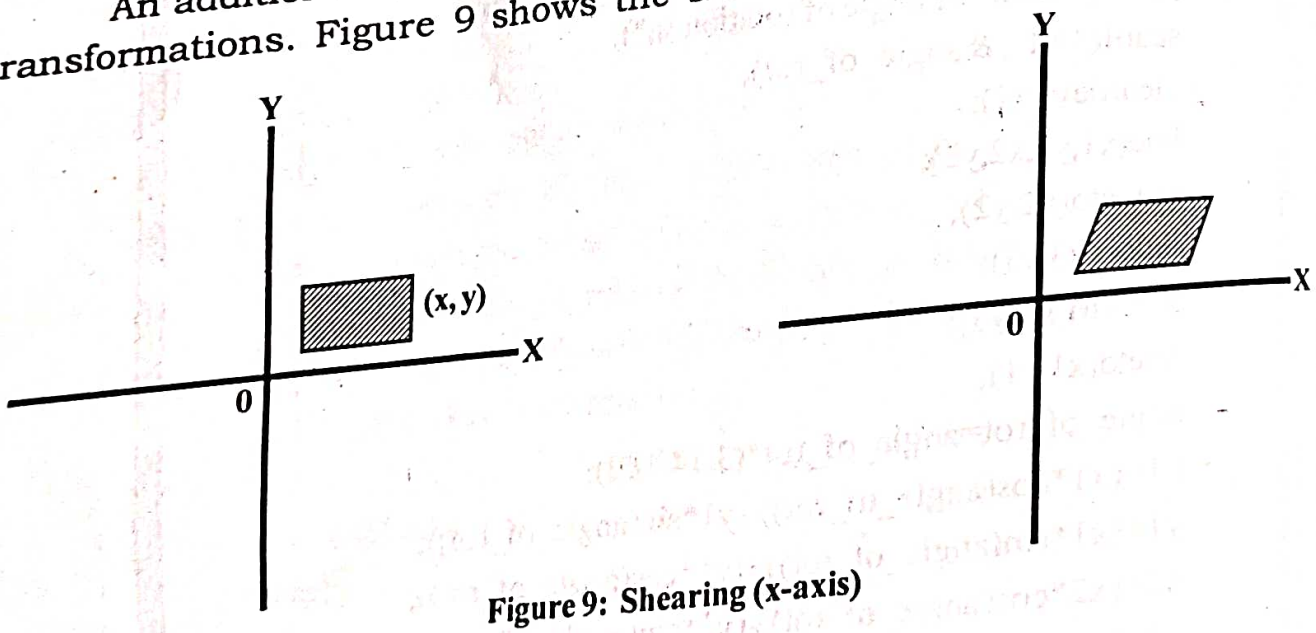

Figure 9: Shearing (x-axis)

The equation for the x direction shear are as follows:

$$x' = x + y \cot \phi$$

$$y' = y$$