INDEX

S.NO	PROGRAM	DATE	SIGNATURE
1.	DDA Line Drawing Algorithm Program		
2.	Mid Point Circle Algorithm Program		
3.	Bresenham’s Circle Algorithm Program		
4.	Translation		
5.	Scaling		
6.	Three Dimentional Transformation		
7.	Simple Animation using Transformation		
8.	Key-Frame Animation		

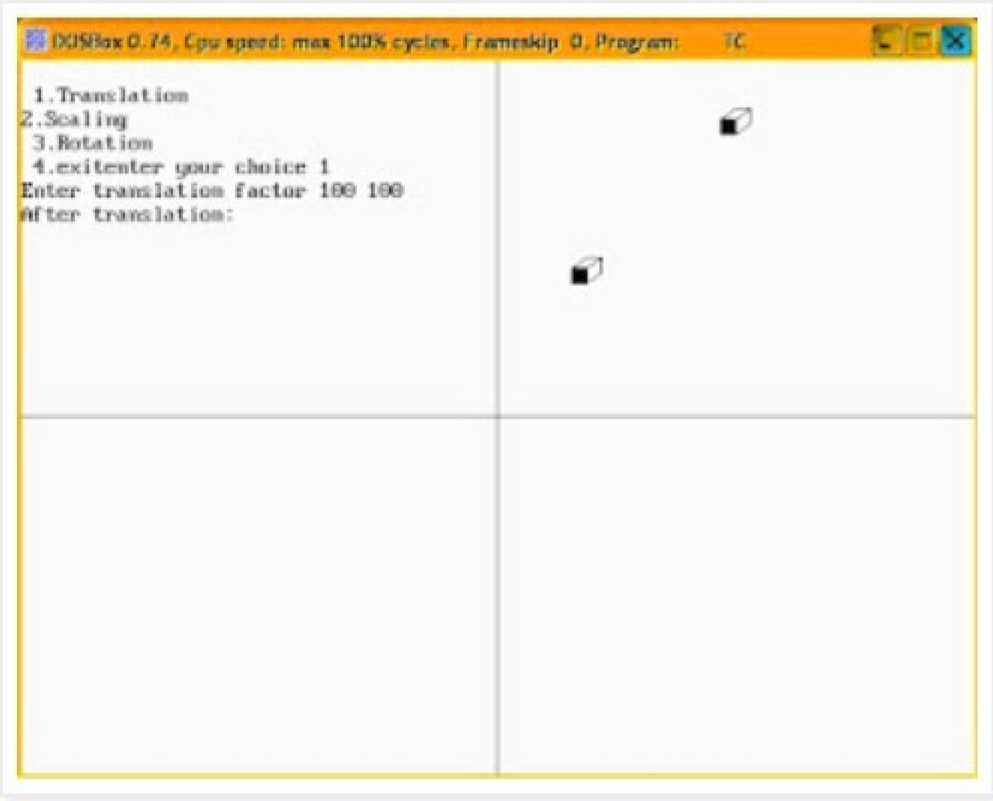
6. 3 D Transformation c Program Code with output.
Code:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void trans();
void scale();
void rotate();
int maxx,maxy,midx,midy;
void main()
{
 int ch;
 int gd=DETECT,gm;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"e:\\tc\\bgi");
printf("\n 1.Translation \n2.Scaling\n 3.Rotation \n 4.exit");
printf("enter your choice");
scanf("%d",&ch);
do
{
switch(ch)
{
case 1 : trans();
getch();
break;
case 2 : scale();
getch();
break;
case 3 :  rotate();
getch();
break;
case 4 :break;
}
printf("enter your choice");
scanf("%d",&ch);
} while(ch<4);
}
void trans()
{
int x,y,z,o,x1,x2,y1,y2;
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
bar3d(midx+50,midy-100,midx+60,midy-90,10,1);
printf("Enter translation factor");
scanf("%d%d",&x,&y);
printf("After translation:");
bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);
}
void scale()
{
int x,y,z,o,x1,x2,y1,y2;
maxx=getmaxx();
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("before translation\n");
printf("Enter scaling factors\n");
scanf("%d %d %d", &x,&y,&z);
printf("After scaling\n");
bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);
}
void rotate()
{
int x,y,z,o,x1,x2,y1,y2;
maxx=getmaxx();
```

```
maxy=getmaxy();
midx=maxx/2;
midy=maxy/2;
bar3d(midx+50,midy-100,midx+60,midy-90,5,1);
printf("Enter rotating angle");
scanf("%d",&o);
x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);
y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);
x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);
y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);
printf("After rotation  about x axis");
bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);
 printf("After rotation about yaxis");
bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);
}
```

Screen short of output:-

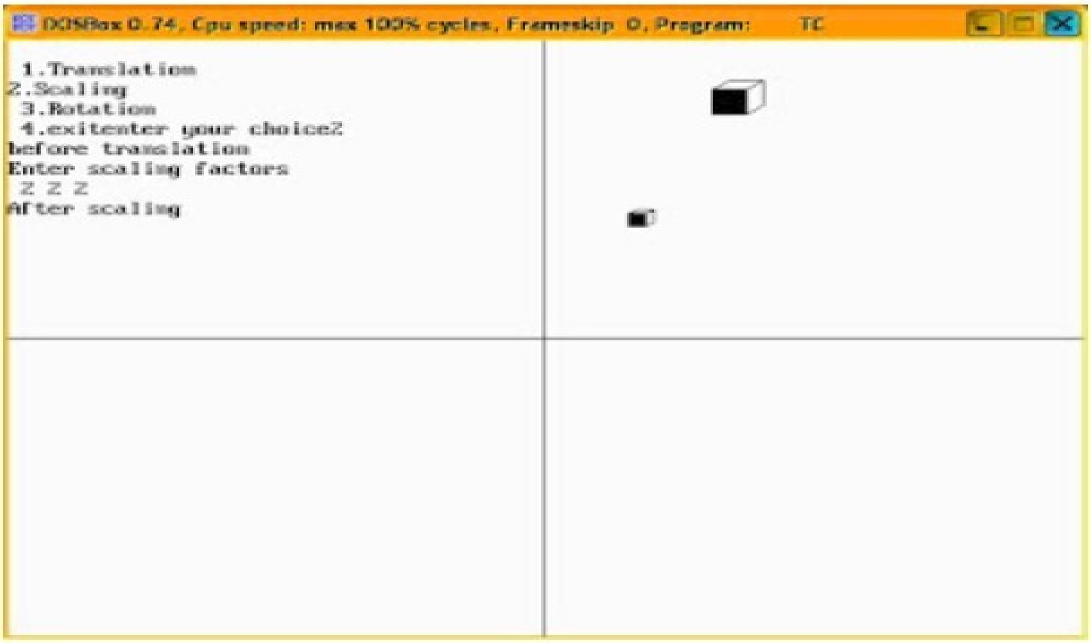
Translation



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

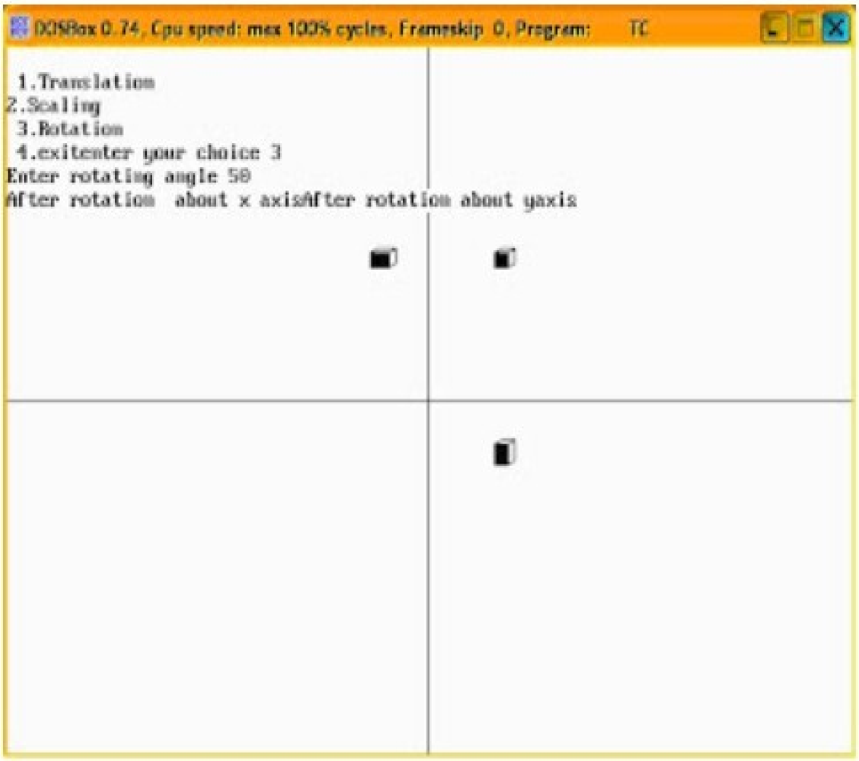
```
 1.Translation
2.Scaling
 3.Rotation
 4.exitenter your choice 1
Enter translation factor 100 100
After translation:
```

Scaling


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 1.Translation
2.Scaling
 3.Rotation
 4.exitenter your choice2
before translation
Enter scaling factors
 2 2 2
After scaling
```

Rotation


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
 1.Translation
2.Scaling
 3.Rotation
 4.exitenter your choice 3
Enter rotating angle 50
After rotation  about x axisAfter rotation about yaxis
```

7. Creating animations using Transformations in c-language

```
#include <stdio.h>
#include <GL/glut.h>
#include <math.h>

// these are the parameters
#define maxHt 800
#define maxWd 600
#define maxLns 10000
#define transSpeed 1
#define rotSpeed 0.02
#define rotateLimit 0.2
#define boundLimitL -200
#define boundLimitR 500
#define grasslandy 230

// Structure for storing lines
typedef struct lines {
        int x1, x2, y1, y2;
} LINE;

// Object type structure for storing each body part
typedef struct objects {
        LINE edge[maxLns];
        int translation, cx, cy, xoffset, yoffset;
        float theta;
        int rotationState;
        int EdgeCount;
} Object;
```

```
// the different objects
Object Head, upBody, Tail, downBody, FlegF, FlegB, BlegF, BlegB;
// global
int dinoTranslate = 0;

// basic init function for OPENGL
void myInit(void)
{
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, maxHt, 0, maxWd);
	glClear(GL_COLOR_BUFFER_BIT);
}

// this function tranlates, and rotates a point according to an object and draws it
void rotateandshiftPt(int px, int py, Object obbj)
{
	int xf, yf;

	xf = obbj.cx + (int)((float)(px - obbj.cx) * cos(obbj.theta)) - ((float)(py - obbj.cy) * sin(obbj.theta));
	yf = obbj.cy + (int)((float)(px - obbj.cx) * sin(obbj.theta)) + ((float)(py - obbj.cy) * cos(obbj.theta));
	glBegin(GL_POINTS);
	glVertex2i(obbj.translation + xf + obbj.xoffset, yf + obbj.yoffset);
	glEnd();
}

// this function draws a line using Bresenhams
void drawLineBresenham(int x1, int y1, int x2, int y2, Object obbj)
```

```
{
    int Dx, Dy, Dxmul2, Dymul2, Pk, xtempi, ytempi;
    float lineSlope, xtemp, ytemp;
    Dx = abs(x2 - x1);
    Dy = abs(y2 - y1);
    Dxmul2 = 2 * Dx;
    Dymul2 = 2 * Dy;
    ytemp = (float)(y2 - y1);
    xtemp = (float)(x2 - x1);
    lineSlope = (ytemp / xtemp);

    if (lineSlope >= -1.0 && lineSlope <= 1.0) {
        Pk = Dymul2 - Dx;
        if (x1 > x2) {
            xtempi = x2;
            x2 = x1;
            x1 = xtempi;
            ytempi = y2;
            y2 = y1;
            y1 = ytempi;
        }
        for (xtempi = x1, ytempi = y1; xtempi <= x2; xtempi++) {
            rotateandshiftPt(xtempi, ytempi, obbj);
            if (Pk < 0) {
                Pk = Pk + Dymul2;
            } else {
                Pk = Pk + Dymul2 - Dxmul2;
                if (lineSlope >= 0.0 && lineSlope <= 1.0)
                    ytempi = ytempi + 1;
                else if (lineSlope < 0.0 && lineSlope >= -1.0)
                    ytempi = ytempi - 1;
            }
        }
    } else {
        Pk = Dxmul2 - Dy;
        if (y1 > y2) {
            xtempi = x2;
            x2 = x1;
            x1 = xtempi;
            ytempi = y2;
            y2 = y1;
            y1 = ytempi;
        }
        for (xtempi = x1, ytempi = y1; ytempi <= y2; ytempi++) {
            rotateandshiftPt(xtempi, ytempi, obbj);
            if (Pk < 0) {
                Pk = Pk + Dxmul2;
            } else {
                Pk = Pk + Dxmul2 - Dymul2;
                if (lineSlope > 1.0)
                    xtempi = xtempi + 1;
                else if (lineSlope < -1.0)
                    xtempi = xtempi - 1;
            }
        }
    }
}
// here all the edges are iterated and drawn
void drawObj(Object obbj)
{
    int i;
    for (i = 0; i < obbj.EdgeCount; i++) {
        drawLineBresenham(obbj.edge[i].x1, obbj.edge[i].y1, obbj.edge[i].x2,
obbj.edge[i].y2, obbj);
```

```
        }
}

// in this function, an object is updated
void updateObj(Object* obbj)
{
        obbj->translation = dinoTranslate;

        if (obbj->rotationState == 1) {
                obbj->theta = obbj->theta + rotSpeed;
                if (obbj->theta >= (3.14159))
                        obbj->theta = obbj->theta - (2.0 * 3.14159);
                if (obbj->theta > rotateLimit)
                        obbj->rotationState = -1;

        } else if (obbj->rotationState == -1) {
                obbj->theta = obbj->theta - rotSpeed;

                if (obbj->theta <= (-3.14159))
                        obbj->theta = (2.0 * 3.14159) + obbj->theta;

                if (obbj->theta < -rotateLimit)
                        obbj->rotationState = 1;
        }
}

// The actual function where the Dinosaur is drawn
void drawDino(void)
{
        // an infinite while loop for moving the dinosaur
        while (1) {
                glClear(GL_COLOR_BUFFER_BIT);
```

```
// draw grassland
glLineWidth(5.0);
glColor3f(0.0f, 1.0f, 0.3f);
glBegin(GL_LINES);
glVertex2i(0, grasslandy);
glVertex2i(maxHt, grasslandy);
glEnd();
glPointSize(3.0);
glColor3f(0.9f, 0.5f, 0.6f);
// update all parts

updateObj(&Head);
updateObj(&upBody);
updateObj(&Tail);
updateObj(&downBody);
updateObj(&FlegF);
updateObj(&FlegB);
updateObj(&BlegF);
updateObj(&BlegB);

// draw all parts, also draw joining parts
drawObj(Head);
drawObj(upBody);
drawObj(Tail);
drawObj(downBody);
drawObj(FlegF);
drawObj(FlegB);
drawObj(BlegF);
drawObj(BlegB);

dinoTranslate--; // decreased because moving forward
if (dinoTranslate <= boundLimitL) {
```

```
            dinoTranslate = boundLimitR;
            printf("\ntranslate %d", dinoTranslate);
        }
        printf("\ntranslate %d", dinoTranslate);
        glFlush();
    }
}

// TAn object is stored using this function
void storeObj(char* str, Object* obbj)
{
    obbj->theta = 0.0;

    FILE* fp;
    fp = fopen(str, "r");
    if (fp == NULL) {
        printf("Could not open file");
        return;
    }
    obbj->EdgeCount = 0;
    int count = 0, x1, y1, x2, y2;
    while (!feof(fp)) {
        count++;
        if (count > 2) {
            x1 = x2;
            y1 = y2;
            count = 2;
        }
        if (count == 1) {
            fscanf(fp, "%d, %d", &x1, &y1);
        } else {
            fscanf(fp, "%d, %d", &x2, &y2);
            printf("\n%d, %d", x2, y2);
            obbj->edge[obbj->EdgeCount].x1 = x1;
            obbj->edge[obbj->EdgeCount].y1 = y1;
            obbj->edge[obbj->EdgeCount].x2 = x2;
            obbj->edge[obbj->EdgeCount].y2 = y2;
            obbj->EdgeCount++;
        }
    }

    // printf("\nPolygon stored!");
    fclose(fp);
}

// All parts are stored.
void storeAllParts()
{
    FILE* fp, *fp2;
    int cx, cy;
    fp = fopen("centrePts.txt", "r");
    fp2 = fopen("offsetDino.txt", "r");
    if (fp == NULL || fp2 == NULL) {
        printf("Could not open file");
        return;
    }
    // parts
    //----------------
    // head+neck
    storeObj("headDino.txt", &Head);
    fscanf(fp, "%d, %d", &cx, &cy);
    Head.cx = cx;
    Head.cy = cy;
    fscanf(fp2, "%d, %d", &cx, &cy);
```

```
Head.xoffset = cx;
Head.yoffset = cy;
Head.rotationState = 1;

// upper body boundary(only translation)
storeObj("bodyupDino.txt", &upBody);
upBody.cx = 0;
upBody.cy = 0;
fscanf(fp2, "%d, %d", &cx, &cy);
upBody.xoffset = cx;
upBody.yoffset = cy;
upBody.rotationState = 0;

// tail
storeObj("tailDino.txt", &Tail);
fscanf(fp, "%d, %d", &cx, &cy);
Tail.cx = cx;
Tail.cy = cy;
fscanf(fp2, "%d, %d", &cx, &cy);
Tail.xoffset = cx;
Tail.yoffset = cy;
Tail.rotationState = -1;

// back leg front
storeObj("backlegFDino.txt", &BlegF);
fscanf(fp, "%d, %d", &cx, &cy);
BlegF.cx = cx;
BlegF.cy = cy;
fscanf(fp2, "%d, %d", &cx, &cy);
BlegF.xoffset = cx;
BlegF.yoffset = cy;
BlegF.rotationState = -1;
```

```
// back leg rear
storeObj("backlegRDino.txt", &BlegB);
fscanf(fp, "%d, %d", &cx, &cy);
BlegB.cx = cx;
BlegB.cy = cy;
fscanf(fp2, "%d, %d", &cx, &cy);
BlegB.xoffset = cx;
BlegB.yoffset = cy;
BlegB.rotationState = 1;

// lower body boundary(only translation)
storeObj("bodydownDino.txt", &downBody);
downBody.cx = 0;
downBody.cy = 0;
fscanf(fp2, "%d, %d", &cx, &cy);
downBody.xoffset = cx;
downBody.yoffset = cy;
downBody.rotationState = 0;

// front leg rear
storeObj("frontlegRDino.txt", &FlegB);
fscanf(fp, "%d, %d", &cx, &cy);
FlegB.cx = cx;
FlegB.cy = cy;
fscanf(fp2, "%d, %d", &cx, &cy);
FlegB.xoffset = cx;
FlegB.yoffset = cy;
FlegB.rotationState = -1;

// front leg front
storeObj("frontlegFDino.txt", &FlegF);
```

```
    fscanf(fp, "%d, %d", &cx, &cy);
    FlegF.cx = cx;
    FlegF.cy = cy;
    fscanf(fp2, "%d, %d", &cx, &cy);
    FlegF.xoffset = cx;
    FlegF.yoffset = cy;
    FlegF.rotationState = 1;

    //------------------------
    fclose(fp);
}

void main(int argc, char** argv)
{

    storeAllParts();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(maxHt, maxWd);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Walking dinosaur");
    myInit();
    glutDisplayFunc(drawDino); // actual loop call
    glutMainLoop();
}
```

8. KeyFraming graphics animation(C program for bouncing ball):-

SourceCode-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>

int main() {
 int gd = DETECT, gm;
 int i, x, y, flag=0;
 initgraph(&gd, &gm, "C:\\TC\\BGI");

 /* get mid positions in x and y-axis */
 x = getmaxx()/2;
 y = 30;


 while (!kbhit()) {
  if(y >= getmaxy()-30 || y <= 30)
     flag = !flag;
     /* draws the gray board */
     setcolor(RED);
     setfillstyle(SOLID_FILL, RED);
     circle(x, y, 30);
     floodfill(x, y, RED);

 /* delay for 50 milli seconds */
 delay(50);

 /* clears screen */
 cleardevice();
 if(flag){
     y = y + 5;
 } else {
     y = y - 5;
 }
    }

    getch();
    closegraph();
    return 0;
}
```

ScreenShort:-