LAB FILE REPORT

Final Record

for

CSE3016-Computer Graphics and Multimedia

Submitted by

<Aryaman Mishra-19BCE1027>

To

DR. S. LINGASWAMY

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

May 2021

Table of contents

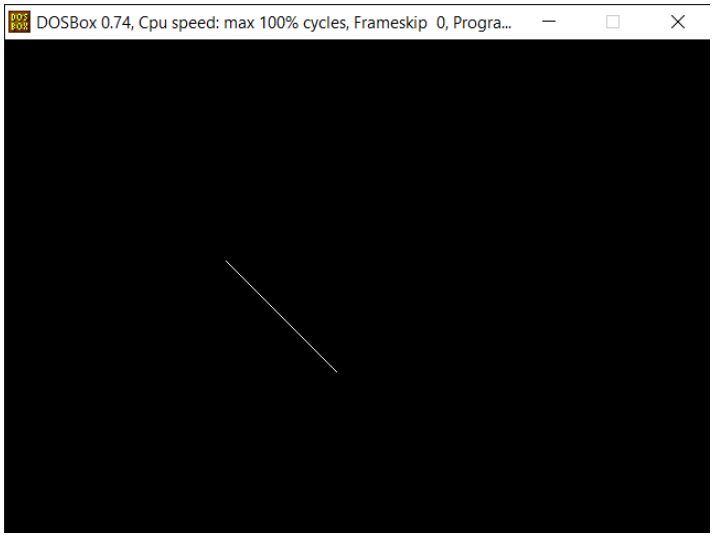
Content Page no.

LAB 1

Multiple Shapes in C(with proof of Code in Turbo C++)

Single Line:

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main(void){

int gdriver=DETECT,gmode;

int x1=200,y1=200;

int x2=300,y2=300;

clrscr();

initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");

line(x1,y1,x2,y2);

getch();

closegraph();

}
```



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

File Edit Search Run Compile Debug Project Options Window Help

\TURBOC3\PROJECTS\LAB1.C

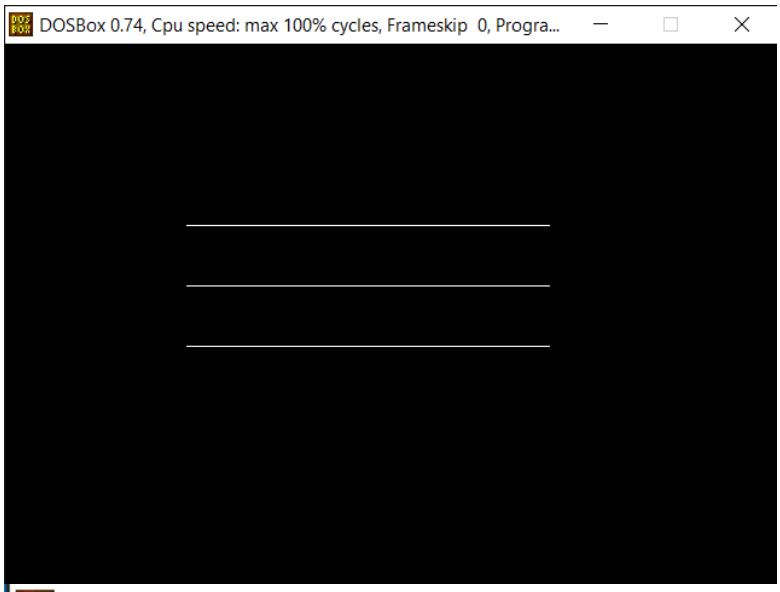
```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main(void){
int gdriver=DETECT,gmode;
int x1=200,y1=200;
int x2=300,y2=300;
clrscr();
initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

1:1

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Multiple Lines:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
line(150,150,450,150);
line(150,200,450,200);
line(150,250,450,250);
getch();
closegraph();
return 0;
}
```



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

File Edit Search Run Compile Debug Project Options Window Help

\TURBOC3\PROJECTS\MULTIPLE.C 1=[↑]

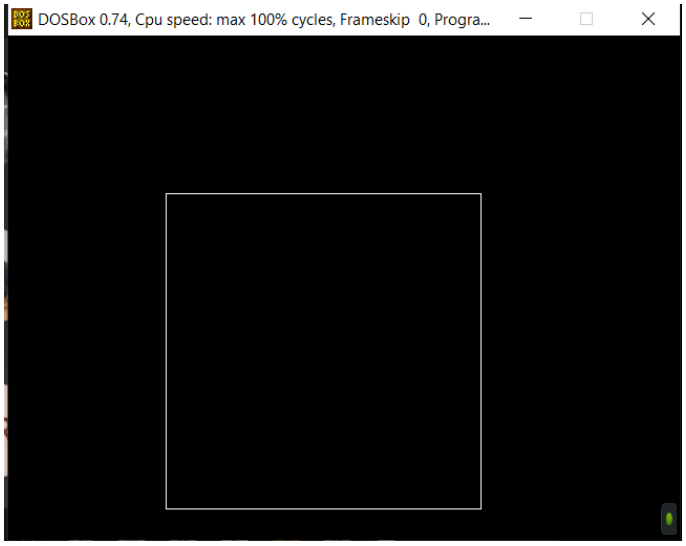
```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
line(150,150,450,150);
line(150,200,450,200);
line(150,250,450,250);
getch();
closegraph();
return 0;
}
```

1:1

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

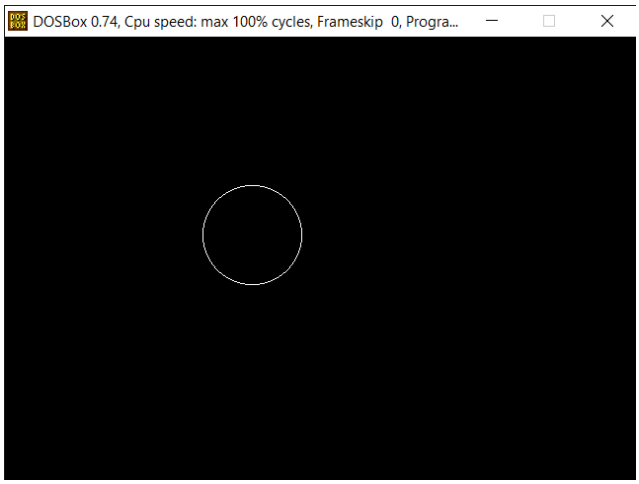
Rectangle:

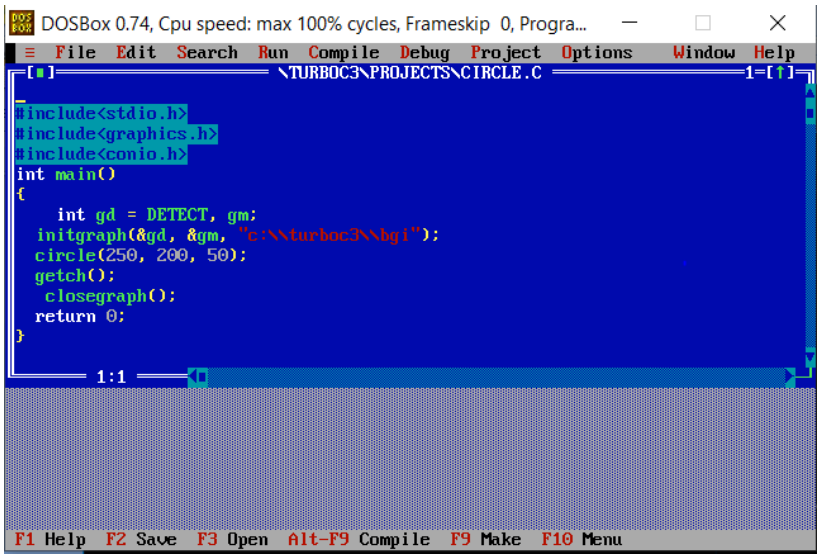
```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{

int gd = DETECT, gm;
  int left = 150, top = 150;
    int right = 450, bottom = 450;
   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
rectangle(left, top, right, bottom);
getch();
  closegraph();
  return 0;
}
```



Circle:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
   int gd = DETECT, gm;
 initgraph(&gd, &gm, "c:\\turboc3\\bgi");
 circle(250, 200, 50);
 getch();
  closegraph();
return 0;}
```

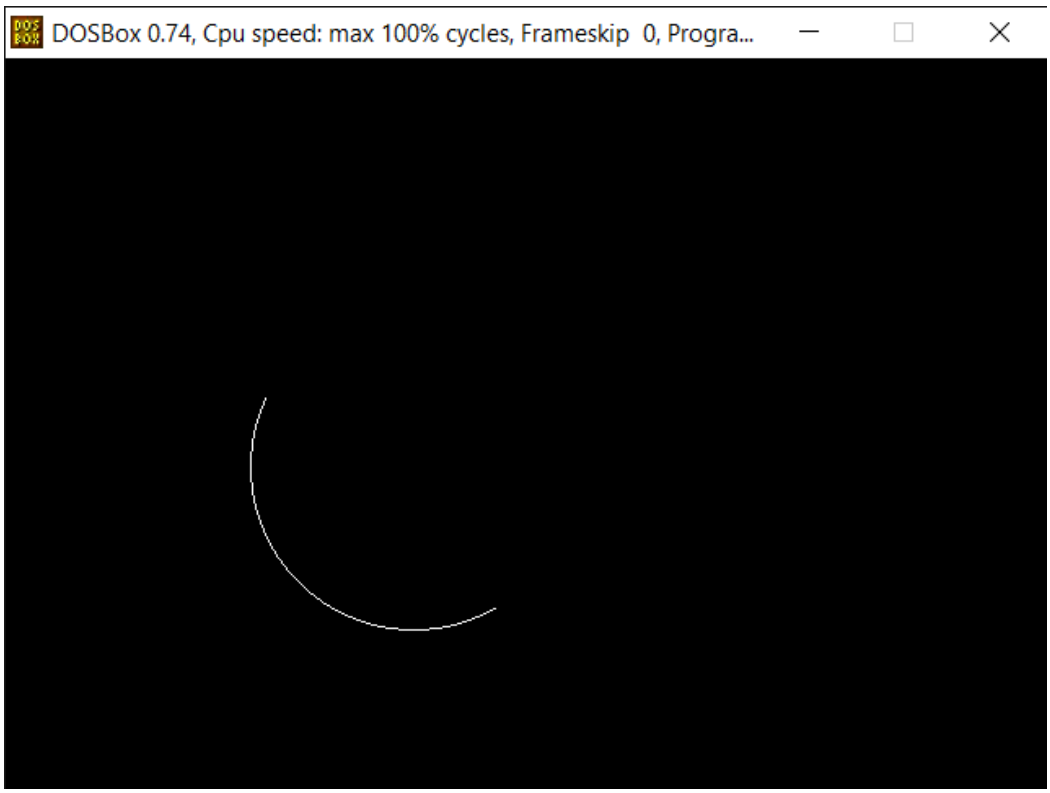


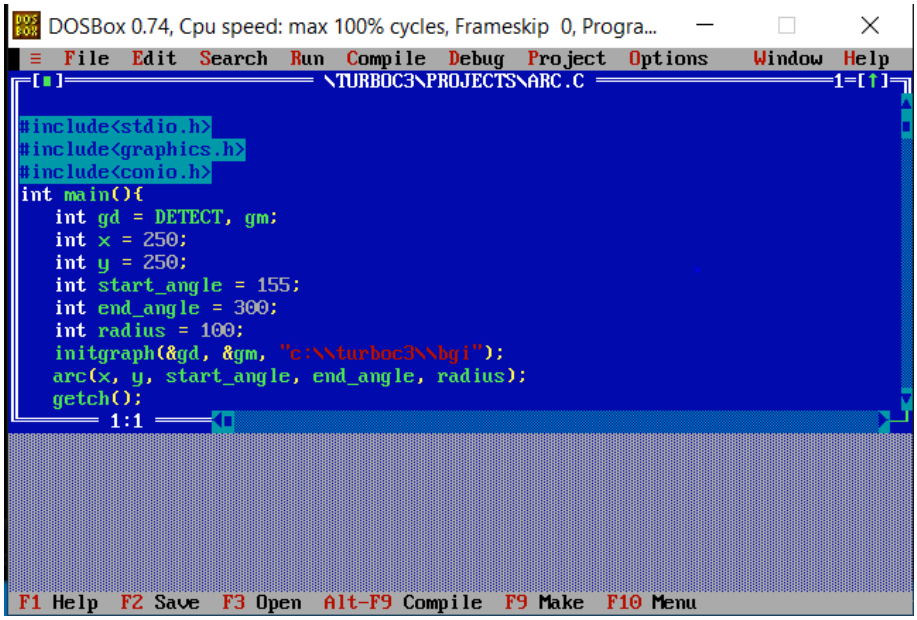


Arc:

```
#include<stdio.h>

#include<graphics.h>

#include<conio.h>

int main(){

  int gd = DETECT, gm;

  int x = 250;

  int y = 250;

  int start_angle = 155;

  int end_angle = 300;

  int radius = 100;

  initgraph(&gd, &gm, "c:\\turboc3\\bgi");

  arc(x, y, start_angle, end_angle, radius);
```

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

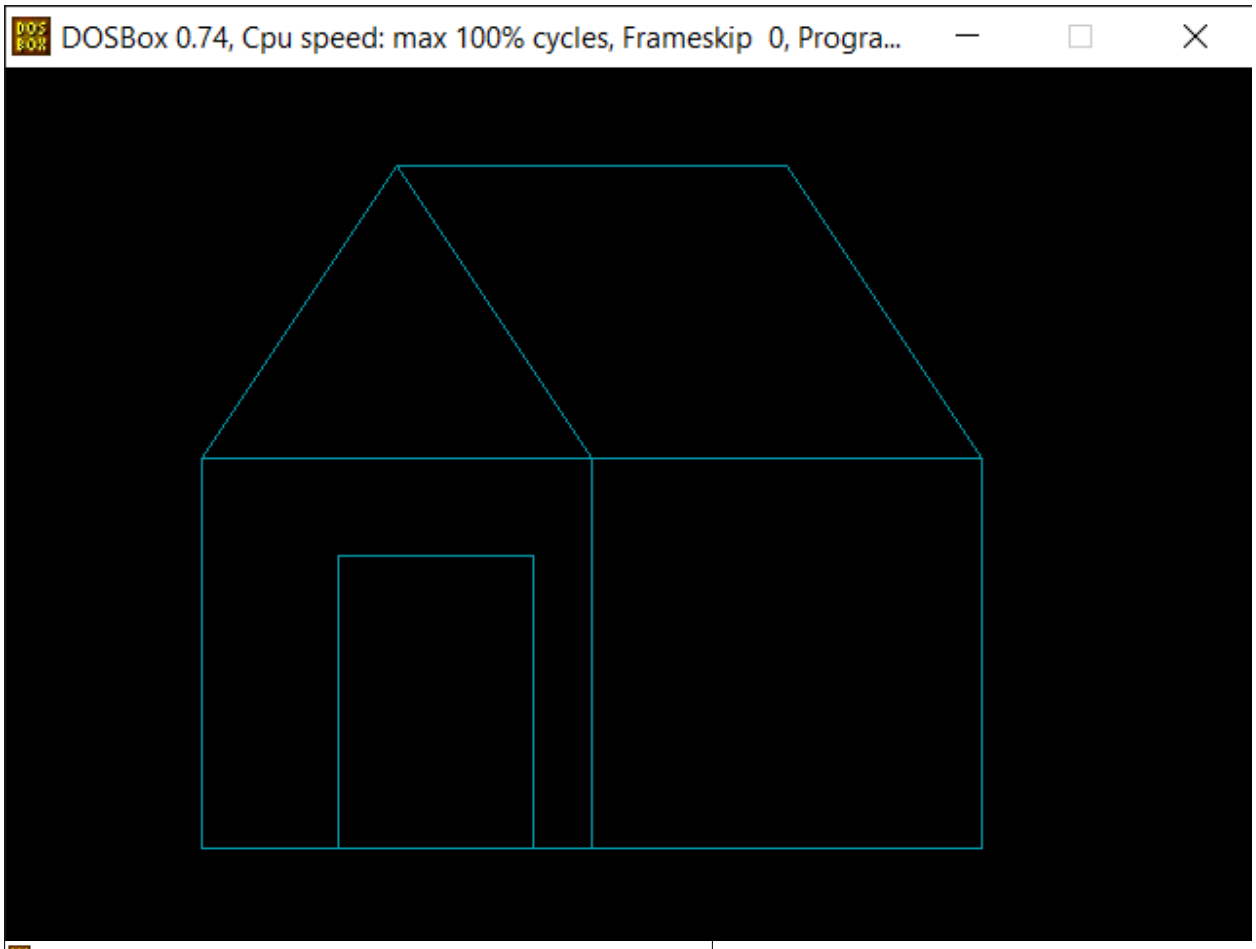


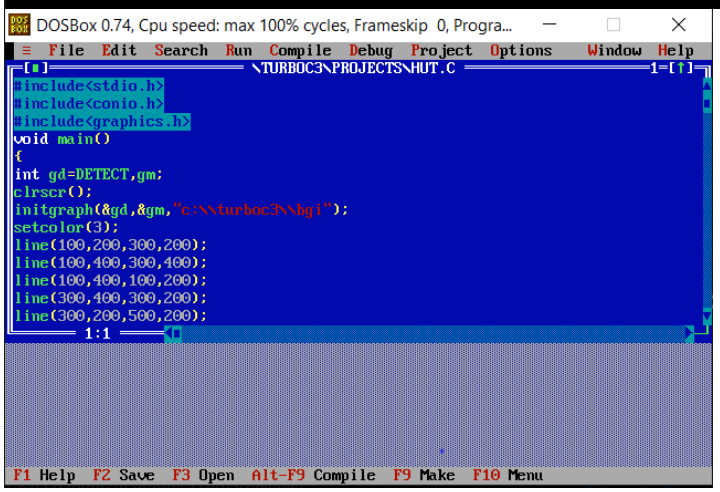


Hut:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

clrscr();

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

setcolor(3);

line(100,200,300,200);

line(100,400,300,400);

line(100,400,100,200);

line(300,400,300,200);
```

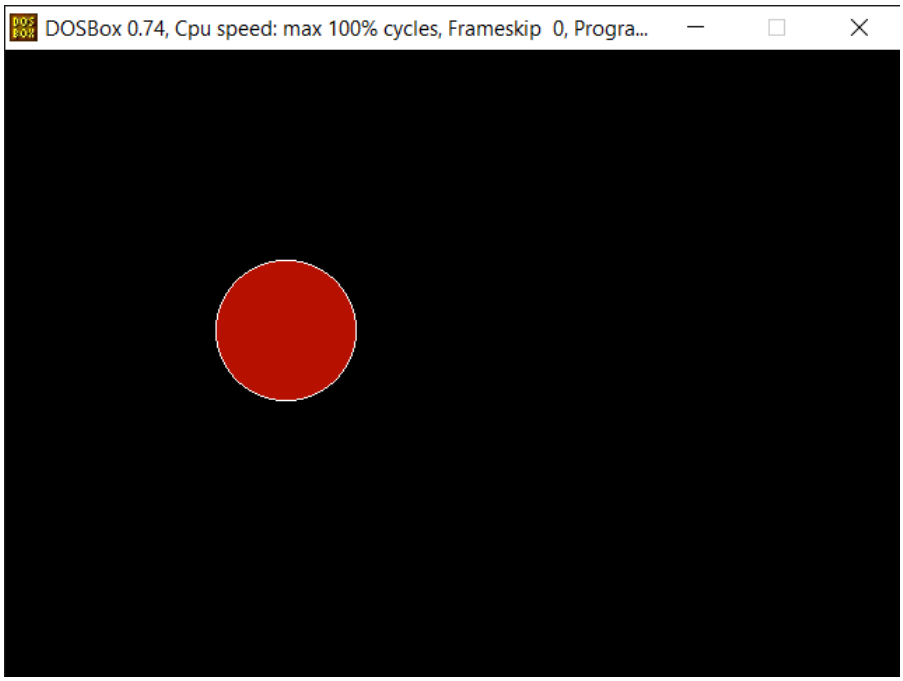
```
line(300,200,500,200);
line(300,400,500,400);
line(500,200,500,400);
line(170,250,270,250);
line(170,200,270,200);
line(170,250,170,400);
line(270,250,270,400);
line(200,50,400,50);
line(400,50,500,200);
line(300,200,200,50);
line(100,200,200,50);
getch();
closegraph();
}
```

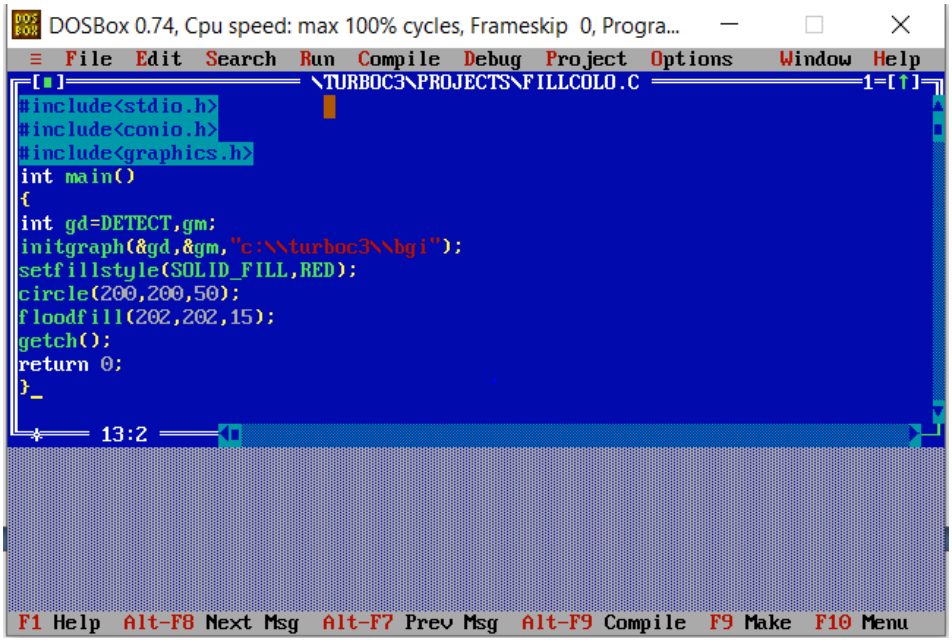

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\HUT.C
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
setcolor(3);
line(100,200,300,200);
line(100,400,300,400);
line(100,400,100,200);
line(300,400,300,200);
line(300,200,500,200);
1:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Fill Color:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int  main()
  {
    int gdriver = DETECT,gmode;
 initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");
 setfillstyle(SOLID_FILL,RED);
 circle(200,200,50);
 floodfill(202,202,15);
 getch();
 return 0;
  }
```




DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\FILLCOLO.C
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
setfillstyle(SOLID_FILL,RED);
circle(200,200,50);
floodfill(202,202,15);
getch();
return 0;
}
13:2
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Lab 2:Scenery,Flag and Object(Rocket)

1.SCENERY

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main()
{
        int gdriver=DETECT,gmode;
        initgraph(&gdriver,&gmode,"C:\\TURBOC3\\BGI");
        line(150,50,200,100);
        line(150,50,80,120);
        line(80,120,100,120);
        line(150,50,350,50);
        line(350,50,400,100);
        line(100,100,100,200);
        line(100,200,200,200);
        line(200,100,200,200);
        line(400,200,400,100);
        line(200,200,400,200);
        line(200,100,400,100);
        rectangle(130,130,170,170);
        rectangle(250,130,320,200);
```

```
line(320,130,305,140);
line(305,140,305,140);
line(250,130,265,140);
line(265,140,265,200);
line(100,200,90,210);
line(90,210,200,200);
line(190,210,200,200);
line(190,210,410,210);
line(400,200,410,200);
//HOUSE COLOR
setfillstyle(8,2);
floodfill(131,131,WHITE);
setfillstyle(11,7);
floodfill(101,101,WHITE);
setfillstyle(1,12);
floodfill(163,55,WHITE);
setfillstyle(1,12);
floodfill(82,119,WHITE);
setfillstyle(3,10);
floodfill(251,121,WHITE);
setfillstyle(1,6);
floodfill(150,205,WHITE);
setfillstyle(1,6);
floodfill(250,205,WHITE);
setfillstyle(5,12);
```

```
floodfill(310,145,WHITE);
setfillstyle(5,12);
floodfill(260,145,WHITE);
//tree
line(505,130,505,200);
line(532,130,532,200);
line(505,200,531,200);
line(480,130,560,130);
line(480,130,500,100);
line(500,100,480,100);
line(480,100,500,70);
line(500,70,480,70);
line(480,70,520,40);
line(560,130,540,100);
line(540,100,560,100);
line(560,100,540,70);
line(540,70,560,70);
line(560,70,520,40);
//color tree
setfillstyle(1,6);
floodfill(506,131,WHITE);
setfillstyle(1,2);
floodfill(510,101,WHITE);
//ROAD
line(270,210,290,390);
```

```
line(315,210,360,390);
line(0,390,290,390);
line(360,390,639,410);
line(0,410,639,410);
line(0,390,0,410);
line(639,390,639,410);
setfillstyle(1,6);
floodfill(1,391,WHITE);
//MOUNTAIN
line(100,180,0,180);
line(400,180,505,180);
line(532,180,639,180);
line(100,150,50,110);
line(50,110,0,150);
line(400,150,450,110);
line(450,110,505,150);
line(532,150,590,110);
line(590,110,639,150);
setfillstyle(1,8);
floodfill(50,50,WHITE);
setfillstyle(1,8);
floodfill(401,150,WHITE);
setfillstyle(1,8);
floodfill(535,150,WHITE);
setfillstyle(1,9);
```

```
        //SKY
                floodfill(0,0,WHITE);
        setfillstyle(1,9);
                floodfill(504,148,WHITE);
        setfillstyle(1,9);
                floodfill(535,132,WHITE);
        //SUN
        circle(70,50,40);
        setfillstyle(1,14);
        floodfill(71,51,WHITE);
        //POND
        ellipse(550,300,0,360,80,50);
        setfillstyle(1,3);
        floodfill(550,330,WHITE);
        //GRASS COLOR
        setfillstyle(1,2);
        floodfill(20,50,WHITE);
        setfillstyle(1,2);
        floodfill(350,230,WHITE);
        getch();

}
```

2.FLAG

```
#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int main()

{

        int gd,gm;

        int r,i,a,b,x,y;

        float PI=3.14;


        detectgraph(&gd,&gm);

        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");


        setcolor(RED);
```

```
rectangle(100,100,450,150);
setfillstyle(SOLID_FILL,RED);
floodfill(101,101,RED);

setcolor(WHITE);
rectangle(100,150,450,200);
setfillstyle(SOLID_FILL,WHITE);
floodfill(101,151,WHITE);

setcolor(GREEN);
rectangle(100,200,450,250);
setfillstyle(SOLID_FILL,GREEN);
floodfill(101,201,GREEN);

a=275;
b=175;
r=25;
setcolor(BLUE);
circle(a,b,r);

//spokes
for(i=0;i<=360;i=i+15)
{
        x=r*cos(i*PI/180);
        y=r*sin(i*PI/180);
```

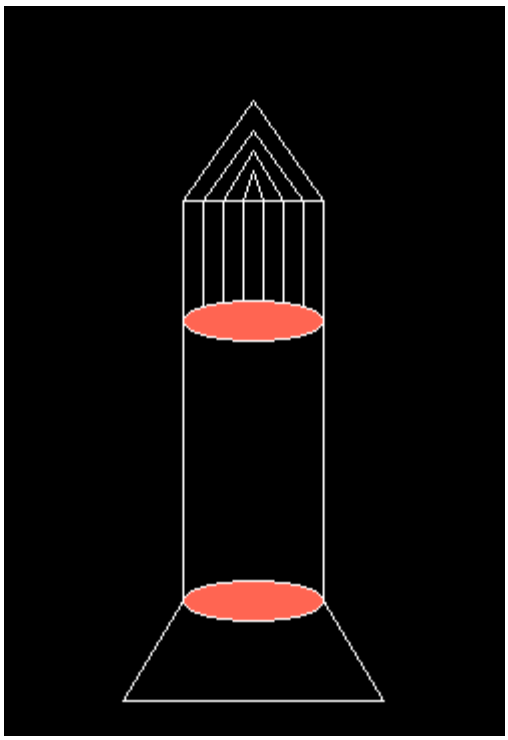
```
            line(a,b,a+x,b-y);
        }

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

3.OBJECT(ROCKET)

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<maths.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm=0;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
line(250,250,250,400);
line(320,250,320,400);
ellipse(285,260,0,360,35,10);
```

```
ellipse(285,400,0,360,35,10);
setfillstyle(SOLID_FILL,12);
fillellipse(285,260,35,10);
fillellipse(285,400,35,10);
setcolor(6);
setcolor(WHITE);
line(250,250,250,200);
line(320,250,320,200);
line(250,200,285,150);
line(320,200,285,150);
line(250,200,320,200);
line(260,200,285,165);
line(310,200,285,165);
line(270,200,285,175);
line(300,200,285,175);
line(280,200,285,185);
line(290,200,285,185);
line(260,200,260,252);
line(270,200,270,251);
line(280,200,280,250);
line(290,200,290,250);
line(300,200,300,251);
line(310,200,310,252);
line(250,400,220,450);
line(320,400,350,450);
```

```
line(220,450,350,450);

setfillstyle(SOLID_FILL,4);

getch();

closegraph();

}
```



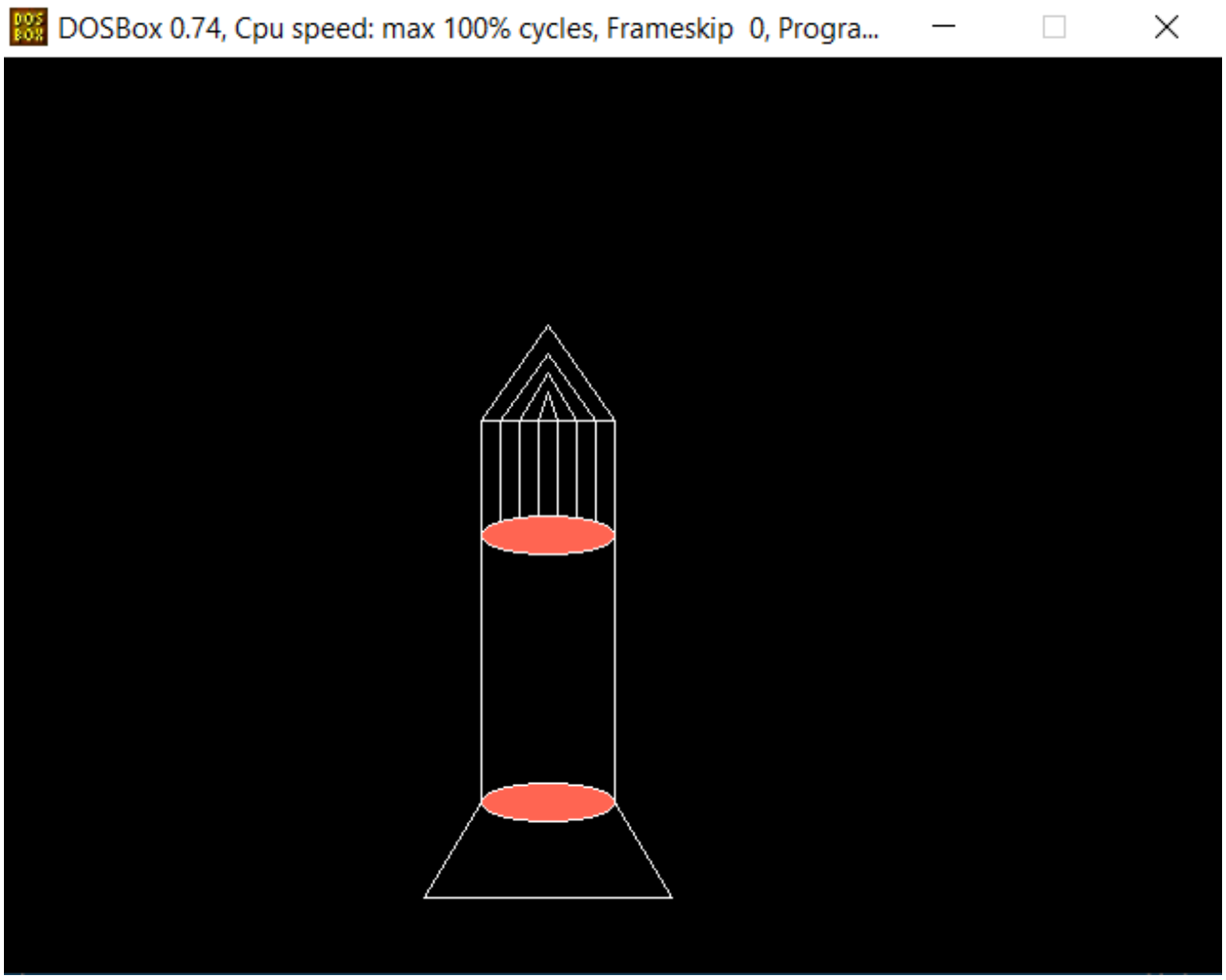
Lab 3:Animation/Moving in TurboC

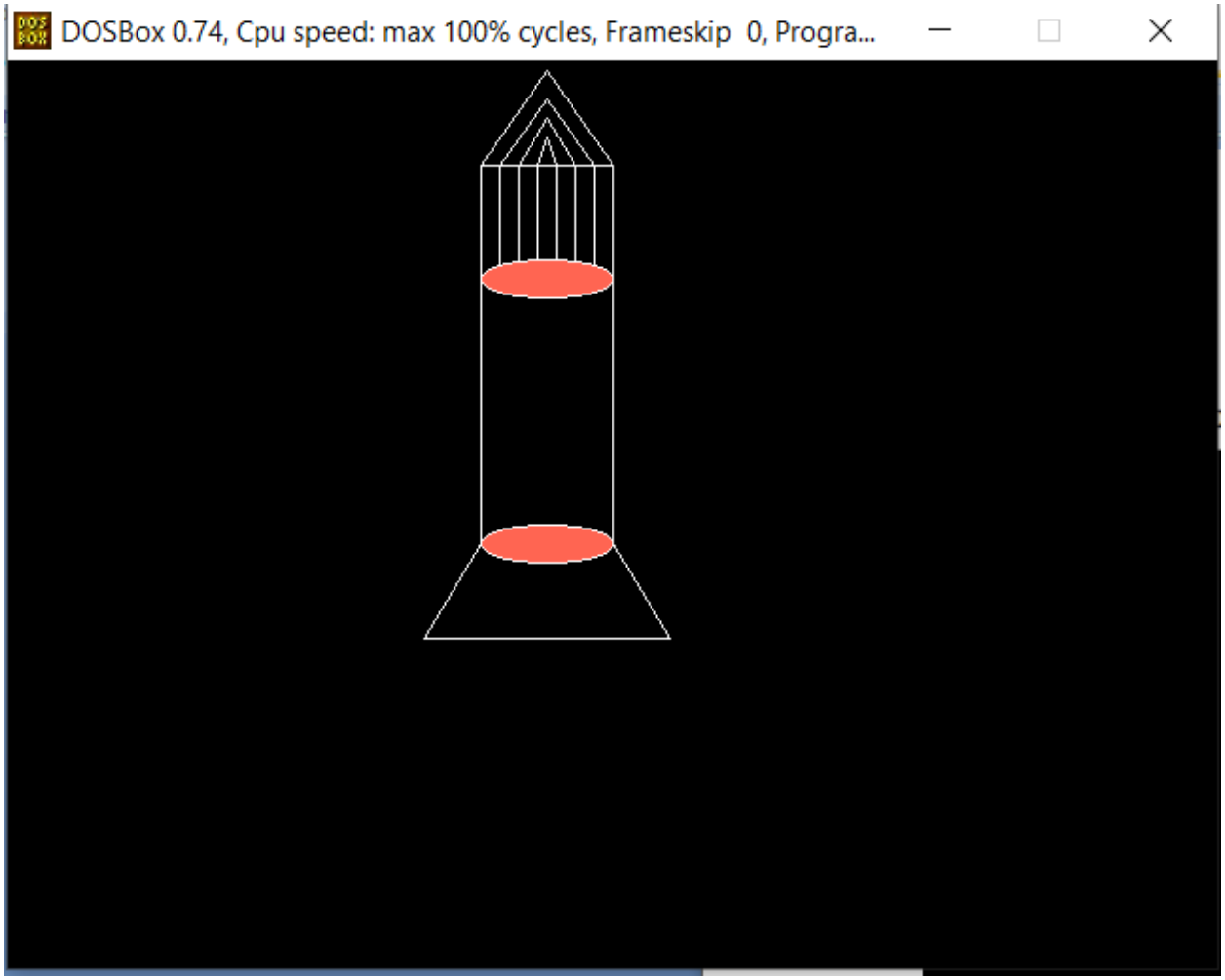
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main()
{
int gd=DETECT,gm=0,i;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
for(i=0;i<300;i++)
{
cleardevice();
line(250,250-i,250,400-i);
line(320,250-i,320,400-i);
ellipse(285,260-i,0,360,35,10);
ellipse(285,400-i,0,360,35,10);
setfillstyle(SOLID_FILL,12);
fillellipse(285,260-i,35,10);
fillellipse(285,400-i,35,10);
setcolor(6);
setcolor(WHITE);
```

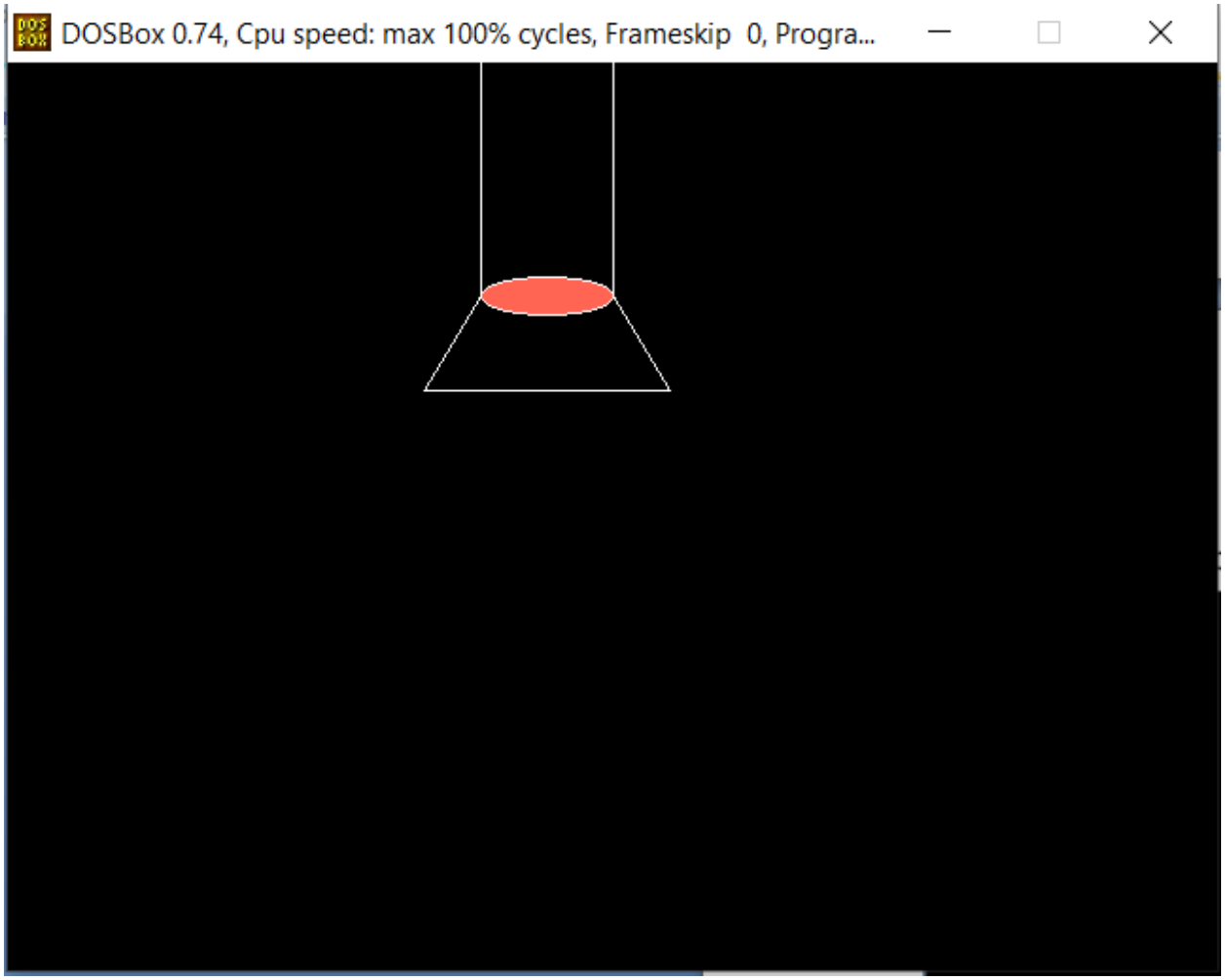
```
line(250,250-i,250,200-i);
line(320,250-i,320,200-i);
line(250,200-i,285,150-i);
line(320,200-i,285,150-i);
line(250,200-i,320,200-i);
line(260,200-i,285,165-i);
line(310,200-i,285,165-i);
line(270,200-i,285,175-i);
line(300,200-i,285,175-i);
line(280,200-i,285,185-i);
line(290,200-i,285,185-i);
line(260,200-i,260,252-i);
line(270,200-i,270,251-i);
line(280,200-i,280,250-i);
line(290,200-i,290,250-i);
line(300,200-i,300,251-i);
line(310,200-i,310,252-i);
line(250,400-i,220,450-i);
line(320,400-i,350,450-i);
line(220,450-i,350,450-i);
delay(200);
}
```

```
getch();
closegraph();
}
```

Starting




DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

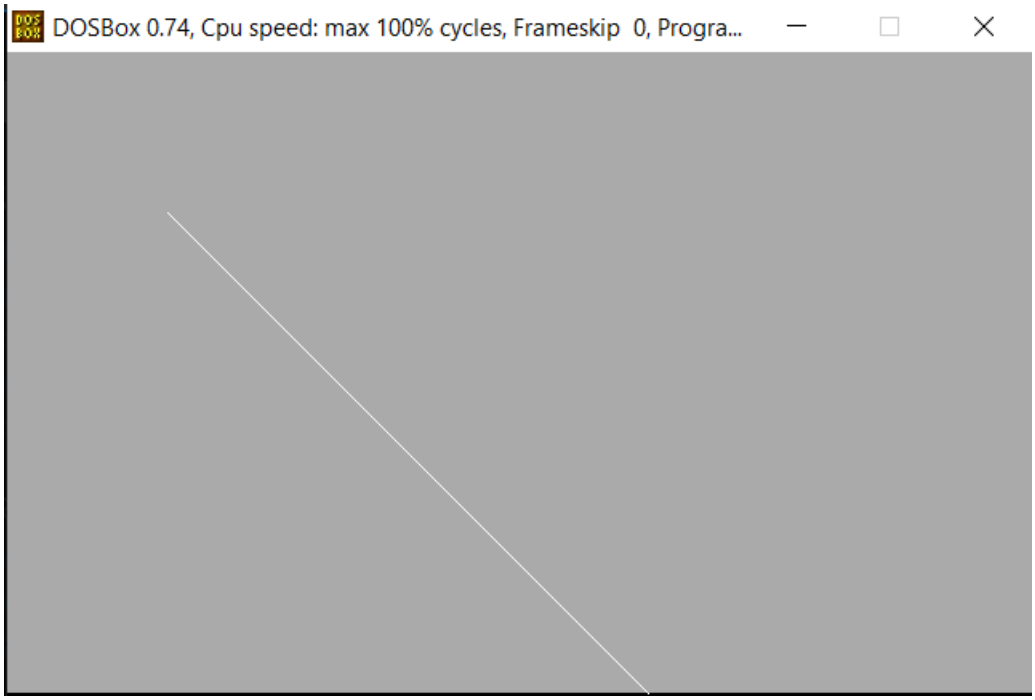

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

Lab 4:Line Drawing Algorithms

Program for DDA Line Drawing Algorithm in C

```
#include<stdio.h>

#include<stdlib.h>

#include<graphics.h>

#include<math.h>


void LineDDA(int x0,int y0,int x1, int y1)

{

double xIncrement,yIncrement,x,y;

int dx = x1 - x0,dy = y1-y0,steps,i;

if(abs(dx)>abs(dy)) steps = abs(dx);

else steps = abs(dy);

//one of these will be 1 or -1

xIncrement = (double)dx/(double)steps;

yIncrement = (double)dy/(double)steps;


 x = x0;

y = y0;

putpixel((int)x,(int)y,WHITE);


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

{
```

```
x+=xIncrement;
y+=yIncrement;
putpixel((int)x,(int)y,WHITE);
}
}

void main()
{
int gd = DETECT,gm;
int x0 = 100,y0 = 100;
int x1 = 400,y1 = 400;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
clrscr();
LineDDA(x0,y0,x1,y1);
getch();
closegraph();
}
```



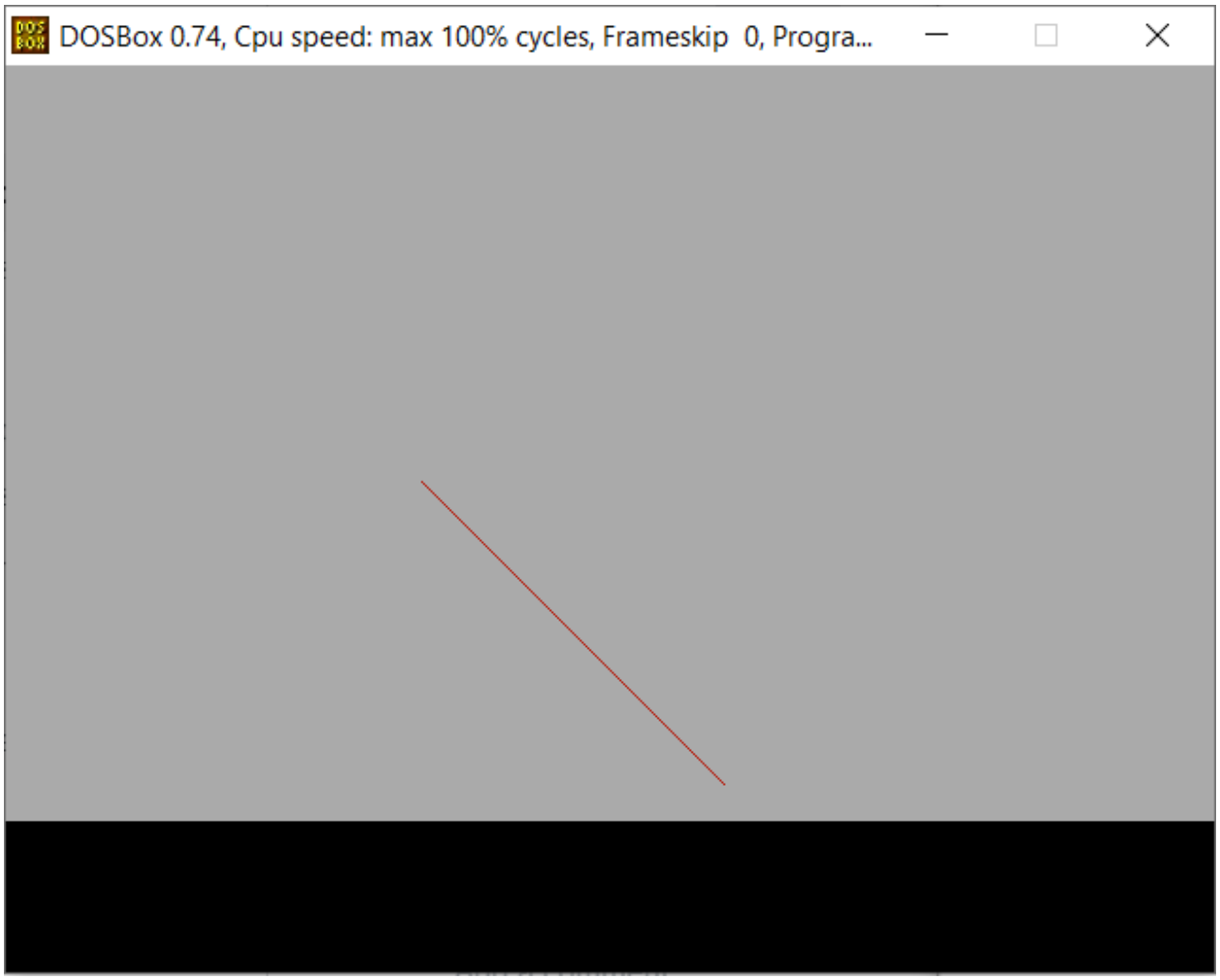
Program for Bresenham's Line Algorithm

Algorithm in C

```
#include<stdio.h>

#include<stdlib.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

void LineBres(int x0,int y0,int x1,int y1)

{

        int dx=abs(x1-x0),dy=abs(y1-y0);

        int d=2*dy-dx,twoDy=2*dy,twoDyMinusDx=2*(dy-dx);

        int x,y;
```

```
if(x0>x1)
{
        x=x1;
        y=y1;
        x1=x0;
}
else
{
        x=x0;
        y=y0;
}
putpixel((int)x,(int)y,RED);
while(x<x1)
{
        x++;
        if(d<0)d+=twoDy;
        else
        {
                y++;
                d+=twoDyMinusDx;
        }
putpixel((int)x,(int)y,RED);
}
```

```
}
void main()
{
        int gd = DETECT,gm;
int x0 = 220,y0 = 220;
int x1 = 380,y1 = 380;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
clrscr();
LineBres(x0,y0,x1,y1);
getch();
closegraph();
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

Lab 5:Circle Drawing and Clipping

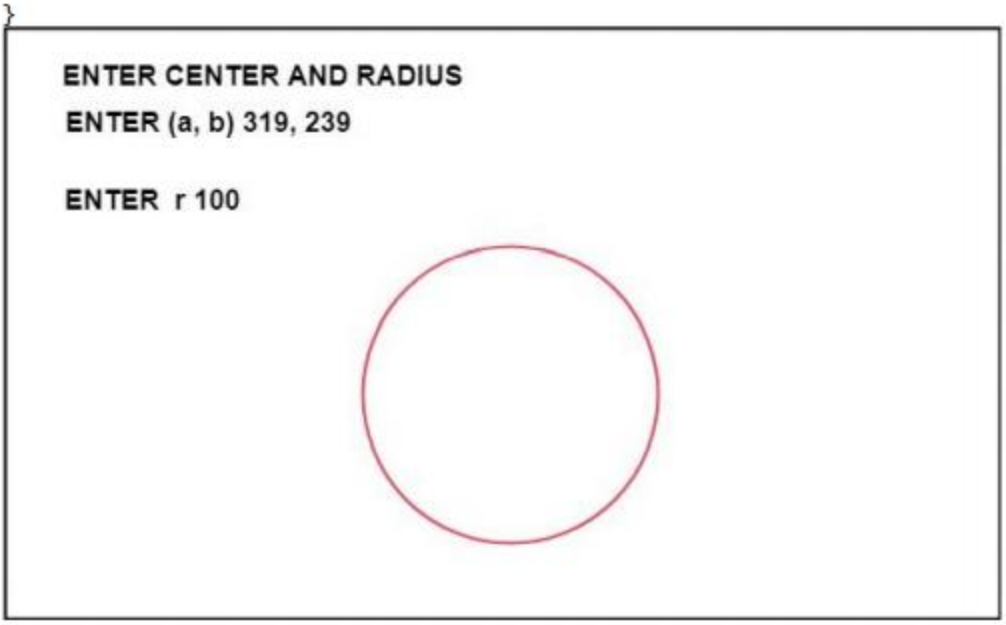
1. Midpoint circle drawing
2. Liang- Barsky Line clipping
3. Weiler Atherton Polygon clipping

Midpoint circle drawing

```
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

class bresen
{
float x, y,a, b, r, p;
public:
void get ();
void cal ();
};
void main ()
{
bresen b;
b.get ();
b.cal ();
getch ();
}
Void bresen :: get ()
{
cout<<"ENTER CENTER AND RADIUS";
cout<< "ENTER (a, b)";
cin>>a>>b;
cout<<"ENTER r";     cin>>r;
}
void bresen ::cal ()
{
```

```
/* request auto detection */
int gdriver = DETECT,gmode, errorcode;
int midx, midy, i;
/* initialize graphics and local variables */
initgraph (&gdriver, &gmode, " ");
/* 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 */
}
x=0;
y=r;
putpixel (a, b+r, RED);
putpixel (a, b-r, RED);
putpixel (a-r, b, RED);
putpixel (a+r, b, RED);
p=5/4)-r;
while (x<=y)
{
If (p<0)
p+= (4*x)+6;
else
{
p+=(2*(x-y))+5;
y--;
}
x++;
putpixel (a+x, b+y, RED);
putpixel (a-x, b+y, RED);
putpixel (a+x, b-y, RED);
putpixel (a+x, b-y, RED);
putpixel (a+x, b+y, RED);
putpixel (a+x, b-y, RED);
```

```
putpixel (a-x, b+y, RED);
putpixel (a-x, b-y, RED);
}
}
```

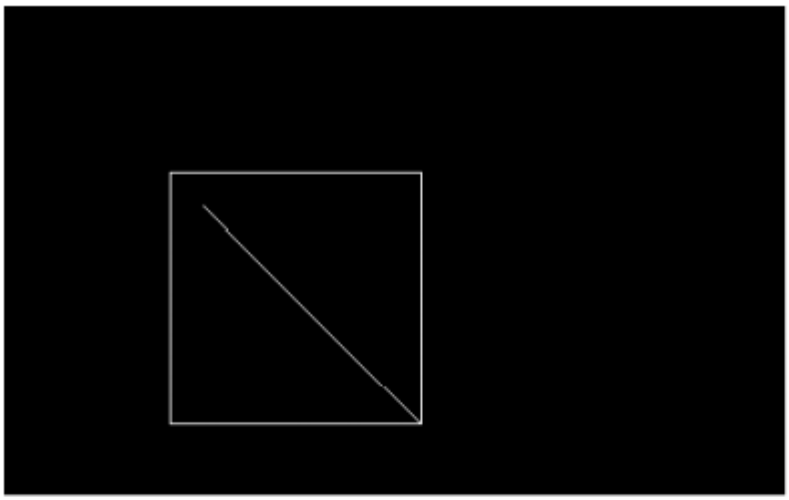


Liang- Barsky Line clipping

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

void main()
{
        int i,gd=DETECT,gm;
        int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
        float t1,t2,p[4],q[4],temp;

        x1=120;
        y1=120;
        x2=300;
        y2=300;

        xmin=100;
        ymin=100;
```

```
xmax=250;
ymax=250;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");
rectangle(xmin,ymin,xmax,ymax);
dx=x2-x1;
dy=y2-y1;

p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;

q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;

for(i=0;i<4;i++)
{
        if(p[i]==0)
        {
                printf("line is parallel to one of the clipping boundary");
                if(q[i]>=0)
                {
                        if(i<2)
                        {
                                if(y1<ymin)
                                {
                                        y1=ymin;
                                }

                                if(y2>ymax)
                                {
                                        y2=ymax;
                                }

                                line(x1,y1,x2,y2);
                        }

                        if(i>1)
                        {
                                if(x1<xmin)
                                {
                                        x1=xmin;
                                }

                                if(x2>xmax)
                                {
                                        x2=xmax;
```

```
                                        }

                                        line(x1,y1,x2,y2);
                                }
                        }
                }
        }

        t1=0;
        t2=1;

        for(i=0;i<4;i++)
        {
                temp=q[i]/p[i];

                if(p[i]<0)
                {
                        if(t1<=temp)
                                t1=temp;
                }
                else
                {
                        if(t2>temp)
                                t2=temp;
                }
        }

        if(t1<t2)
        {
                xx1 = x1 + t1 * p[1];
                xx2 = x1 + t2 * p[1];
                yy1 = y1 + t1 * p[3];
                yy2 = y1 + t2 * p[3];
                line(xx1,yy1,xx2,yy2);
        }

        delay(5000);
        closegraph();
}
```



Weiler Atherton Polygon clipping

```
#include<iostream>
                    #include<graphics.h>
                    #include<utility>
                    using namespace std;
                    struct vertex
                    {
                       float x;
                       float y;
                    };
                    vertex cw[40], sp[40];
                    int n_cw, n_sp;
                    void draw_poly(vertex vlist[], int n)
                    {
                       for (int i=0; i<n; i++)
                          line(vlist[i].x, vlist[i].y, vlist[(i+1)%n].x, vlist[(i+1)%n].y);
                    }
                    int in_out(float x, float y, int x1, int y1, int x2, int y2)
                    {
                       float p = (y-y1)*(x2-x1) - (x-x1)*(y2-y1);
                       if (p<0)
                          return 0;  //for out
                       return 1;    //for in
                    }
                    void intersection_lineseg(float &x, float &y, int x1, int y1, int x2,
```

```
int y2, int xa, int ya, int xb, int yb)
{
   x = -1;
   y = -1;
   if (x2==x1 && xb==xa)
      return;
   else if (x2==x1)
   {
      float m2 = (yb-ya) / (float) (xb-xa);
      x = x1;
      y = ya - m2*(xa-x1);
   }
   else if (xb==xa)
   {
      float m1 = (y2-y1) / (float) (x2-x1);
      x = xa;
      y = y1 + m1*(xa-x1);
   }
   else
   {
      float m1 = (y2-y1) / (float) (x2-x1);
      float m2 = (yb-ya) / (float) (xb-xa);
      if (m1==m2)
         return;
      x = (ya-y1 + m1*x1 - m2*xa) / (m1-m2);
      y = (m1*m2*(xa-x1) + m2*y1 - m1*ya) / (m2-m1);
   }
   if ((x1>=x2 && (x<x2||x>x1)) || (x2>=x1 && (x>x2||x<x1)) ||
(y1>=y2 && (y<y2||y>y1)) || (y2>=y1 && (y>y2||y<y1))
         || (xa>=xb && (x<xb||x>xa)) || (xb>=xa &&
(x>xb||x<xa)) || (ya>=yb && (y<yb||y>ya)) || (yb>=ya &&
(y>yb||y<ya)))
   {
      x = -1;
      y = -1;
   }
}
void wa_clip()
{
   vertex tempcw[40], tempsp[40];
   int tag_sp[40], tag_cw[40], trav_sp[40], trav_cw[40];
   float x,y;
   int entry_list[10];      //saves indexes only
   int e=-1;
   //for new cw array
   int kc=-1;   //first vertex gets added last in array
```

```
   for (int i=0; i<n_cw; i++)
   {
      vertex tempi[20][2];          //for ordering intersection
points, 2nd column's x is for tag
      int ti = -1;
      for (int j=0; j<n_sp; j++)
      {
         intersection_lineseg(x, y, cw[i].x, cw[i].y,
cw[(i+1)%n_cw].x, cw[(i+1)%n_cw].y,
                              sp[j].x, sp[j].y, sp[(j+1)%n_sp].x,
sp[(j+1)%n_sp].y);
         if (x==-1)  //or y==-1
            continue;
         ti++;
         tempi[ti][0].x = x;
         tempi[ti][0].y = y;
         int p1 = in_out(sp[j].x, sp[j].y, cw[i].x, cw[i].y,
cw[(i+1)%n_cw].x, cw[(i+1)%n_cw].y);
         int p2 = in_out(sp[(j+1)%n_sp].x, sp[(j+1)%n_sp].y,
cw[i].x, cw[i].y, cw[(i+1)%n_cw].x, cw[(i+1)%n_cw].y);
         if (p1==1 && p2==0)
            tempi[ti][1].x = 1;
         else
            tempi[ti][1].x = 0;
      }
      if (ti!=-1)
      {
         if (cw[(i+1)%n_cw].x > cw[i].x)          //sort
intersection points
         {
            //increasing x sort
            int min_idx;
            for (int k=0; k<ti; k++)
            {
               min_idx = k;
               for (int m=k+1; m<ti+1; m++)
               {
                  if (tempi[m][0].x < tempi[min_idx][0].x)
                     min_idx=m;
               }
               float temp = tempi[min_idx][0].x;
               tempi[min_idx][0].x = tempi[k][0].x;
               tempi[k][0].x = temp;
               temp = tempi[min_idx][0].y;
               tempi[min_idx][0].y = tempi[k][0].y;
               tempi[k][0].y = temp;
```

```
            temp = tempi[min_idx][1].x;
            tempi[min_idx][1].x = tempi[k][1].x;
            tempi[k][1].x = temp;
        }
    }
    else if (cw[(i+1)%n_cw].x < cw[i].x)
    {
        //decreasing x sort
        int max_idx;
        for (int k=0; k<ti; k++)
        {
            max_idx = k;
            for (int m=k+1; m<ti+1; m++)
            {
                if (tempi[m][0].x > tempi[max_idx][0].x)
                    max_idx=m;
            }
            float temp = tempi[max_idx][0].x;
            tempi[max_idx][0].x = tempi[k][0].x;
            tempi[k][0].x = temp;
            temp = tempi[max_idx][0].y;
            tempi[max_idx][0].y = tempi[k][0].y;
            tempi[k][0].y = temp;
            temp = tempi[max_idx][1].x;
            tempi[max_idx][1].x = tempi[k][1].x;
            tempi[k][1].x = temp;
        }
    }
    else if (cw[(i+1)%n_cw].y > cw[i].y)
    {
        //increasing y sort
        int min_idx;
        for (int k=0; k<ti; k++)
        {
            min_idx = k;
            for (int m=k+1; m<ti+1; m++)
            {
                if (tempi[m][0].y < tempi[min_idx][0].y)
                    min_idx=m;
            }
            float temp = tempi[min_idx][0].x;
            tempi[min_idx][0].x = tempi[k][0].x;
            tempi[k][0].x = temp;
            temp = tempi[min_idx][0].y;
            tempi[min_idx][0].y = tempi[k][0].y;
            tempi[k][0].y = temp;
```

```
                temp = tempi[min_idx][1].x;
                tempi[min_idx][1].x = tempi[k][1].x;
                tempi[k][1].x = temp;
            }
        }
        else
        {
            //decreasing y sort
            int max_idx;
            for (int k=0; k<ti; k++)
            {
                max_idx = k;
                for (int m=k+1; m<ti+1; m++)
                {
                    if (tempi[m][0].y > tempi[max_idx][0].y)
                        max_idx=m;
                }
                float temp = tempi[max_idx][0].x;
                tempi[max_idx][0].x = tempi[k][0].x;
                tempi[k][0].x = temp;
                temp = tempi[max_idx][0].y;
                tempi[max_idx][0].y = tempi[k][0].y;
                tempi[k][0].y = temp;
                temp = tempi[max_idx][1].x;
                tempi[max_idx][1].x = tempi[k][1].x;
                tempi[k][1].x = temp;
            }
        }
        for (int k=0; k<=ti; k++)                    //put sorted
intersection points in cw array
        {
            kc++;
            tempcw[kc].x = tempi[k][0].x;
            tempcw[kc].y = tempi[k][0].y;
            tag_cw[kc] = tempi[k][1].x;
            trav_cw[kc] = 0;
        }
    }
    kc++;
    tempcw[kc].x = cw[(i+1)%n_cw].x;
    tempcw[kc].y = cw[(i+1)%n_cw].y;
    tag_cw[kc] = -1;
    trav_cw[kc] = 0;
}
//for new sp array
int ks=-1;    //first vertex gets added last in array
```

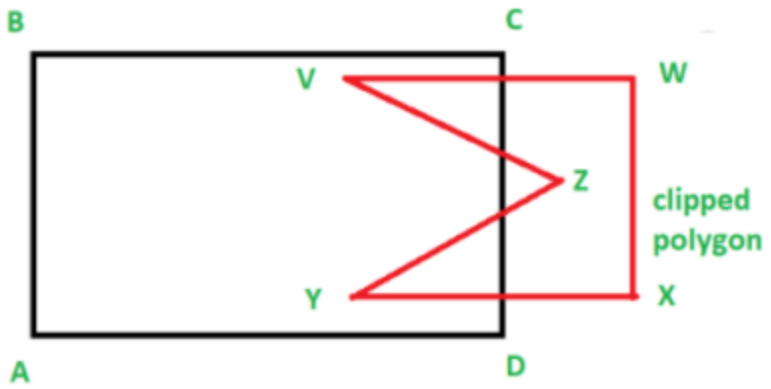
```
   for (int i=0; i<n_sp; i++)
   {
      vertex tempi[20][2];          //for ordering intersection
points, 2nd column's x is for tag
      int ti = -1;
      for (int j=0; j<n_cw; j++)
      {
         intersection_lineseg(x, y, cw[j].x, cw[j].y,
cw[(j+1)%n_cw].x, cw[(j+1)%n_cw].y,
                              sp[i].x, sp[i].y, sp[(i+1)%n_sp].x,
sp[(i+1)%n_sp].y);
         if (x==-1)  //or y==-1
            continue;
         ti++;
         tempi[ti][0].x = x;
         tempi[ti][0].y = y;
         int p1 = in_out(sp[i].x, sp[i].y, cw[j].x, cw[j].y,
cw[(j+1)%n_cw].x, cw[(j+1)%n_cw].y);
         int p2 = in_out(sp[(i+1)%n_sp].x, sp[(i+1)%n_sp].y,
cw[j].x, cw[j].y, cw[(j+1)%n_cw].x, cw[(j+1)%n_cw].y);
         if (p1==1 && p2==0) {
            tempi[ti][1].x = 0;
            }
         else {
            tempi[ti][1].x = 1;
         }
      }
      if (ti!=-1)
      {
         if (sp[(i+1)%n_sp].x > sp[i].x)          //sort intersection
points
         {
            //increasing x sort
            int min_idx;
            for (int k=0; k<ti; k++)
            {
               min_idx = k;
               for (int m=k+1; m<ti+1; m++)
               {
                  if (tempi[m][0].x < tempi[min_idx][0].x)
                     min_idx=m;
               }
               float temp = tempi[min_idx][0].x;
               tempi[min_idx][0].x = tempi[k][0].x;
               tempi[k][0].x = temp;
               temp = tempi[min_idx][0].y;
```

```
            tempi[min_idx][0].y = tempi[k][0].y;
            tempi[k][0].y = temp;
            temp = tempi[min_idx][1].x;
            tempi[min_idx][1].x = tempi[k][1].x;
            tempi[k][1].x = temp;
        }
    }
    else if (sp[(i+1)%n_sp].x < sp[i].x)
    {
        //decreasing x sort
        int max_idx;
        for (int k=0; k<ti; k++)
        {
            max_idx = k;
            for (int m=k+1; m<ti+1; m++)
            {
                if (tempi[m][0].x > tempi[max_idx][0].x)
                    max_idx=m;
            }
            float temp = tempi[max_idx][0].x;
            tempi[max_idx][0].x = tempi[k][0].x;
            tempi[k][0].x = temp;
            temp = tempi[max_idx][0].y;
            tempi[max_idx][0].y = tempi[k][0].y;
            tempi[k][0].y = temp;
            temp = tempi[max_idx][1].x;
            tempi[max_idx][1].x = tempi[k][1].x;
            tempi[k][1].x = temp;
        }
    }
    else if (sp[(i+1)%n_sp].y > sp[i].y)
    {
        //increasing y sort
        int min_idx;
        for (int k=0; k<ti; k++)
        {
            min_idx = k;
            for (int m=k+1; m<ti+1; m++)
            {
                if (tempi[m][0].y < tempi[min_idx][0].y)
                    min_idx=m;
            }
            float temp = tempi[min_idx][0].x;
            tempi[min_idx][0].x = tempi[k][0].x;
            tempi[k][0].x = temp;
            temp = tempi[min_idx][0].y;
```

```
                tempi[min_idx][0].y = tempi[k][0].y;
                tempi[k][0].y = temp;
                temp = tempi[min_idx][1].x;
                tempi[min_idx][1].x = tempi[k][1].x;
                tempi[k][1].x = temp;
            }
        }
        else
        {
            //decreasing y sort
            int max_idx;
            for (int k=0; k<ti; k++)
            {
                max_idx = k;
                for (int m=k+1; m<ti+1; m++)
                {
                    if (tempi[m][0].y > tempi[max_idx][0].y)
                        max_idx=m;
                }
                float temp = tempi[max_idx][0].x;
                tempi[max_idx][0].x = tempi[k][0].x;
                tempi[k][0].x = temp;
                temp = tempi[max_idx][0].y;
                tempi[max_idx][0].y = tempi[k][0].y;
                tempi[k][0].y = temp;
                temp = tempi[max_idx][1].x;
                tempi[max_idx][1].x = tempi[k][1].x;
                tempi[k][1].x = temp;
            }
        }
        for (int k=0; k<=ti; k++)                    //put sorted
intersection points in sp array
        {
            ks++;
            tempsp[ks].x = tempi[k][0].x;
            tempsp[ks].y = tempi[k][0].y;
            tag_sp[ks] = tempi[k][1].x;
            if (tag_sp[ks]==1) {
                e++;
                entry_list[e] = ks;
            }
            trav_sp[ks] = 0;
        }
    }
    ks++;
    tempsp[ks].x = sp[(i+1)%n_sp].x;
```

```
            tempsp[ks].y = sp[(i+1)%n_sp].y;
            tag_sp[ks] = -1;
            trav_sp[ks] = 0;
        }
        n_cw = kc+1;
        n_sp = ks+1;
        //traversal
        for (int i=0; i<=e; i++)
        {
            bool done = false;
            int j = entry_list[i];
            while(!done)
            {
                if (trav_sp[j] == 1)
                    done = true;
                else if (tag_sp[j] == 1 || tag_sp[j] == -1)
                {
                    line(tempsp[j].x, tempsp[j].y, tempsp[(j+1)%n_sp].x,
tempsp[(j+1)%n_sp].y);
                    trav_sp[j] = 1;
                    j++;
                }
                else if (tag_sp[j] == 0)
                {
                    trav_sp[j] = 1;
                    //swap
                    for (int k=0; k<n_cw; k++)     //find location to switch
to
                    {
                        if (tempcw[k].x==tempsp[j].x &&
tempcw[k].y==tempsp[j].y)
                        {
                            j = k;
                            break;
                        }
                    }
                    swap(tempcw, tempsp);
                    swap(tag_cw, tag_sp);
                    swap(trav_cw, trav_sp);
                    int n = n_cw;
                    n_cw = n_sp;
                    n_sp = n_cw;
                }
            }
        }
    }
```

```
int main()
{
   int gd=DETECT, gm=0;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   cout<<"Enter no. of vertices in clipping window"<<endl;
   cin>>n_cw;
   cout<<"Enter vertices (x,y) clockwise"<<endl;
   for (int i=0; i<n_cw; i++)
      cin>>cw[i].x>>cw[i].y;
   draw_poly(cw, n_cw);
   cout<<"Enter no. of vertices in subject polygon"<<endl;
   cin>>n_sp;
   cout<<"Enter vertices (x,y) clockwise"<<endl;
   for (int i=0; i<n_sp; i++)
      cin>>sp[i].x>>sp[i].y;
   draw_poly(sp, n_sp);
   char ch;
   cout<<"Press a key to clip"<<endl;
   cin>>ch;
   cleardevice();
   draw_poly(cw, n_cw);
   wa_clip();
   getch();
   closegraph();
   return 0;
}
```



Lab 6:2D Transformation

```
#include <stdio.h>
#include<graphics.h>
#include<stdlib.h>
#include<conio.h>
#include<math.h>
void DrawFn();
void translate();
int graDriver=DETECT,graMode;
int n,xs[100],ys[100],i,xshift,yshift;
void DrawFn()
{
for(i=0;i<n;i++)
 {
 line(xs[i],ys[i],xs[(i+1)%n],ys[(i+1)%n]);
 }
}
void translate()
{
for(i=0;i<n;i++)
 {
 xs[i]+=xshift;
 ys[i]+=yshift;
 }
```

```
}
void refx(int x1,int x2,int x3,int y1,int y2,int y3){
line(320,0,320,430);
line(0,240,640,240);
x1=(320-x1)+320;
x2=(320-x2)+320;
x3=(320-x3)+320;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);

}
void refy(int x1,int x2,int x3,int y1,int y2,int y3){
line(320,0,320,430);
line(0,240,640,240);
y1=(240-y1)+240;
y2=(240-y2)+240;
y3=(240-y3)+240;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
}
void findNewCoordinate(int s[][2], int p[][1])
{
```

```
	int temp[2][1] = { 0 };

	for (int i = 0; i < 2; i++)
		for (int j = 0; j < 1; j++)
			for (int k = 0; k < 2; k++)
				temp[i][j] += (s[i][k] * p[k][j]);

	p[0][0] = temp[0][0];
	p[1][0] = temp[1][0];
}
void scale(int x[], int y[], int sx, int sy)
{
	// Triangle before Scaling
	line(x[0], y[0], x[1], y[1]);
	line(x[1], y[1], x[2], y[2]);
	line(x[2], y[2], x[0], y[0]);

	// Initializing the Scaling Matrix.
	int s[2][2] = { sx, 0, 0, sy };
	int p[2][1];

	// Scaling the triangle
	for (int i = 0; i < 3; i++)
	{
```

```
                p[0][0] = x[i];
                p[1][0] = y[i];

                findNewCoordinate(s, p);

                x[i] = p[0][0];
                y[i] = p[1][0];
            }

            // Triangle after Scaling
            line(x[0], y[0], x[1], y[1]);
            line(x[1], y[1], x[2], y[2]);
            line(x[2], y[2], x[0], y[0]);
}
int main () {
  char choice;
  printf("Enter 1 for translation,2 for reflection,3 for rotation,4 for scaling,5 for shearing along x
axis,6 for shearing along y axis.\n");
  scanf("%c", &choice)

  switch(choice) {
    case '1' :
      int graDriver=DETECT,graMode;
int n,xs[100],ys[100],i,xshift,yshift;
```

```
printf("Enter number of sides of polygon: ");
scanf("%d",&n);
printf("Enter co-rdinates: x,y for each vertex ");
for(i=0;i<n;i++)
 scanf("%d%d",&xs[i],&ys[i]);
printf("Enter distances for translation (in x and y directions): ");
scanf("%d%d",&xshift,&yshift);
initgraph(&graDriver,&graMode,"C:\\TURBOC3\\BGI\\");
cleardevice();
//drawing original polygon in RED color
setcolor(RED);
DrawFn();
//Doing translation
translate();
//drawing translated polygon in BLUE color
setcolor(BLUE);
DrawFn();
getch();
break;
case '2' :
int gd=DETECT,gm;
int x1,y1,x2,y2,x3,y3;
clrscr();
initgraph(&gd,&gm,"c://turboc3//bgi");
```

```
line(320,0,320,430);
line(0,240,640,240);
x1=150;y1=100;
x2=220;y2=220;
x3=220;y3=110;
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
getch();
refx(x1,x2,x3,y1,y2,y3);
getch();
refy(x1,x2,x3,y1,y2,y3);
getch();
closegraph();
break;
     case '3' :
          intgd=0,gm,x1,y1,x2,y2,x3,y3;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
   scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
   setbkcolor(WHITE);
   cleardevice();
```

```
line(x1,y1,x2,y2);
line(x2,y2, x3,y3);
line(x3, y3, x1, y1);
getch();
setbkcolor(BLACK);
printf("Enter rotation angle: ");
scanf("%lf", &angle);
setbkcolor(WHITE);
c = cos(angle *M_PI/180);
s = sin(angle *M_PI/180);
x1 = floor(x1 * c + y1 * s);
y1 = floor(-x1 * s + y1 * c);
x2 = floor(x2 * c + y2 * s);
y2 = floor(-x2 * s + y2 * c);
x3 = floor(x3 * c + y3 * s);
y3 = floor(-x3 * s + y3 * c);
cleardevice();
line(x1, y1 ,x2, y2);
line(x2,y2, x3,y3);
line(x3, y3, x1, y1);
getch();
closegraph();
  case '4' :
  int x[] = { 100, 200, 300 };
```

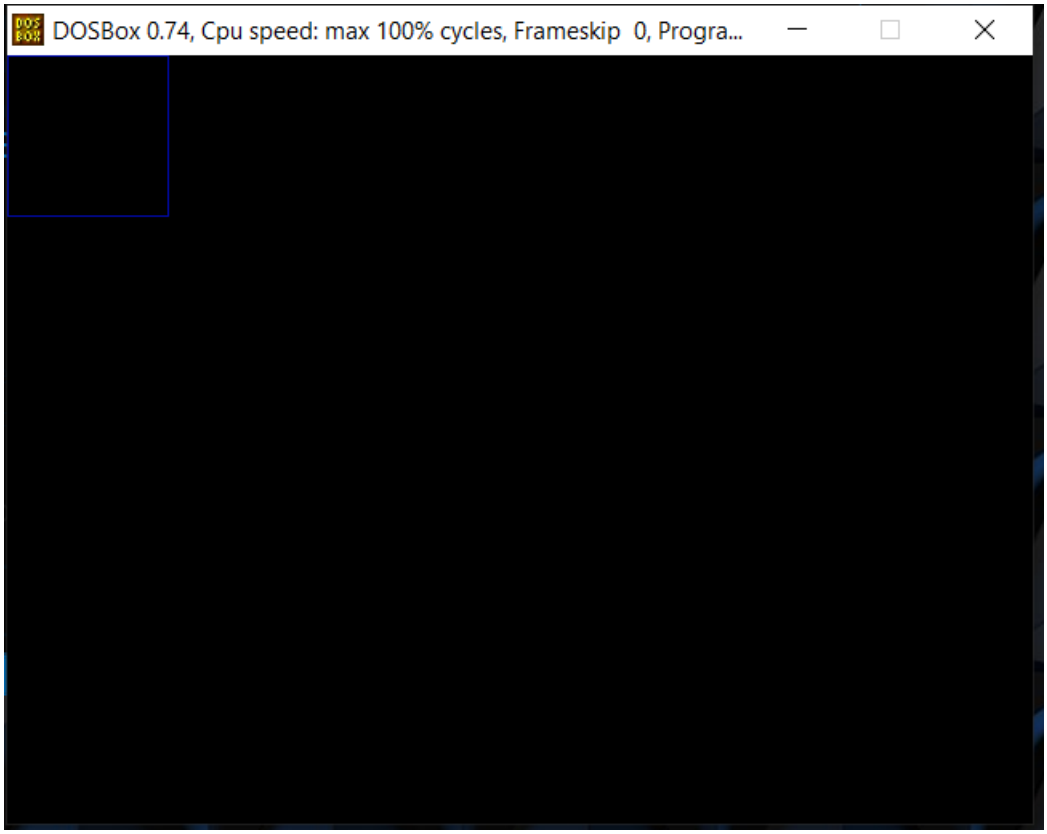
```
            int y[] = { 200, 100, 200 };
            int sx = 2, sy = 2;

            int gd, gm;
            detectgraph(&gd, &gm);
            initgraph(&gd, &gm," ");

            scale(x, y, sx,sy);
            getch();
          break;
       case '5' :int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,x3,y3,shear_f;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = ");
scanf("%d %d",&x2,&y2);
printf("\n please enter last coordinate = ");
scanf("%d %d",&x3,&y3);
printf("\n please enter shearing factor x = ");
scanf("%d",&shear_f);
cleardevice();
```

```
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x,y);

setcolor(RED);
x=x+ y*shear_f;
x1=x1+ y1*shear_f;
x2=x2+ y2*shear_f;
x3=x3+ y3*shear_f;

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x,y);
getch();
closegraph();
        break;
        case '6':int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,x3,y3,shear_f;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
```

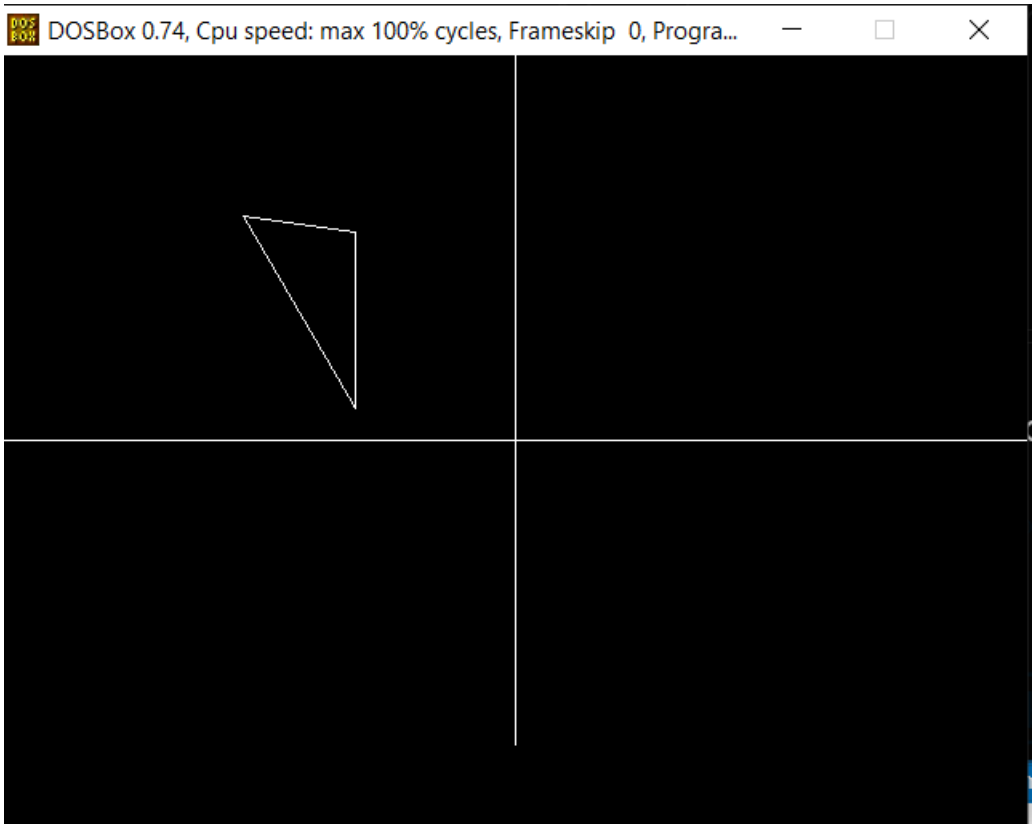
```
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = ");
scanf("%d %d",&x2,&y2);
printf("\n please enter last coordinate = ");
scanf("%d %d",&x3,&y3);
printf("\n please enter shearing factor y = ");
scanf("%d",&shear_f);
cleardevice();
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x,y);

setcolor(RED);
y=y+ x*shear_f;
y1=y1+ x1*shear_f;
y2=y2+ x2*shear_f;
y3=y3+ x3*shear_f;

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x,y);
getch();
```

```
closegraph();
break;
    default :
      printf("Wrong Choice.Try Again.\n" );
  }
  return 0;
}
```

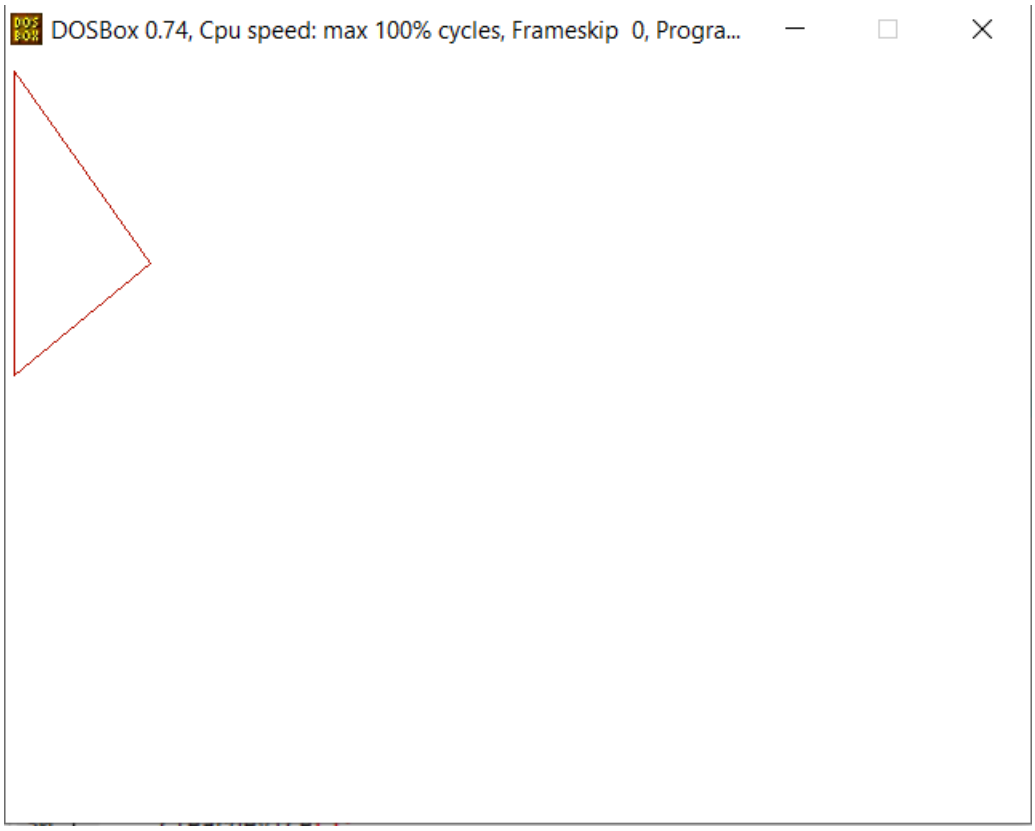
TRANSLATION:



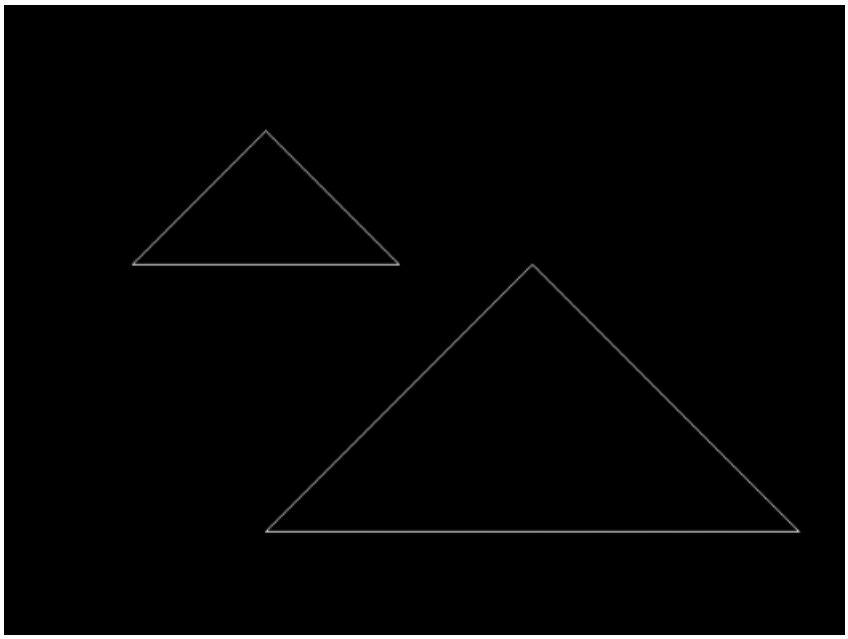
REFLECTION:



ROTATION:

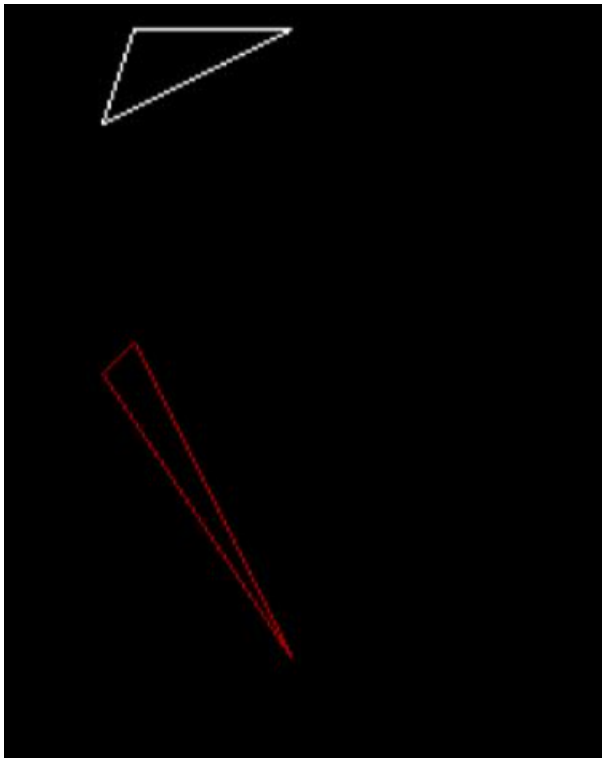


SCALING:



SHEARING ALONG X-AXIS:

SHEARING ALONG Y-AXIS:



Lab 7:3D Transformation

```
#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <stdlib.h>

#include <dos.h>
```

```
#include <conio.h>
#define ORG -50
# define  f              0.3
 # define  projection_angle   45
 void show_screen( );

 void apply_x_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);

 void draw_pyramid(constint [5][3]);
 void get_projected_point(int&,int&,int&);

 void Line(constint,constint,constint,constint);
  void show_screen( );

 void apply_y_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);

void trans();
#define ORG -50

double face1[5][2] = {
      { 250, 125 },
      { 350, 125 },
```

```
      { 350, 225 },
      { 250, 225 },
      { 250, 125 }
           };

double face2[5][2] = {
      { 250+ORG, 125-ORG },
      { 350+ORG, 125-ORG },
      { 350+ORG, 225-ORG },
      { 250+ORG, 225-ORG },
      { 250+ORG, 125-ORG }
           };

double angle = 5.0 * M_PI / 180;
double midx1, midy1, midx2, midy2;

void rotate (void)
{
   int i;
   for (i=0; i<5; i++)
   {
 double xnew, ynew;

 xnew = midx1 + (face1[i][0] - midx1) * cos (angle) -
```

```
      (face1[i][1] - midy1) * sin (angle);
ynew = midy1 + (face1[i][0] - midx1) * sin (angle) +
 (face1[i][1] - midy1) * cos (angle);

face1[i][0] = xnew;
face1[i][1] = ynew;

xnew = midx2 + (face2[i][0] - midx2) * cos (angle) -
 (face2[i][1] - midy2) * sin (angle);
ynew = midy2 + (face2[i][0] - midx2) * sin (angle) +
 (face2[i][1] - midy2) * cos (angle);

face2[i][0] = xnew;
face2[i][1] = ynew;
  }

  cleardevice();

  for (i=0; i<4; i++)
  {
setcolor(7);
line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);
setcolor(8);
line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);
```

```
 setcolor(9);
 line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);
   }

   delay (125);
}
void apply_y_shearing(int edge_points[5][3],constfloat a,constfloat b)
   {
     for(int count=0;count<5;count++)
     {
       float matrix_a[4]={edge_points[count][0],edge_points[count][1],
                   edge_points[count][2],1};
       float matrix_b[4][4]={
              { 1,0,0,0 } ,
              { a,1,b,0 } ,
              { 0,0,1,0 } ,
              { 0,0,0,1 }
             };

       float matrix_c[4]={0};

       multiply_matrices(matrix_a,matrix_b,matrix_c);

       edge_points[count][0]=(int)(matrix_c[0]+0.5);
```

```
        edge_points[count][1]=(int)(matrix_c[1]+0.5);
        edge_points[count][2]=(int)(matrix_c[2]+0.5);
     }
   }

//these are left,top,right,bottom parameters for bar3d function
int maxx,maxy,midx,midy;
//function for translation of a 3d object
void trans()

{

          int x,y,z,o,x1,x2,y1,y2;

          midx=200;

          midy=200;

          //function to draw 3D rectangular box
          bar3d(midx+50,midy-100,midx+100,midy-50,20,1);
  delay(1000);
          printf("Enter translation factor");

          scanf("%d%d",&x,&y);
```

```
        printf("After translation:");

        bar3d(midx+x+50,midy-(y+100),midx+x+100,midy-(y+50),20,1);

}
int x1,x2,y1,y2,mx,my,depth;
void draw();
void rotate();
void rotate()
{
   float t;
   int a1,b1,a2,b2,dep;
   printf("Enter the angle to rotate=");
   scanf("%f",&t);
   t=t*(3.14/180);
   a1=mx+(x1-mx)*cos(t)-(y1-my)*sin(t);
   a2=mx+(x2-mx)*cos(t)-(y2-my)*sin(t);
   b1=my+(x1-mx)*sin(t)-(y1-my)*cos(t);
   b2=my+(x2-mx)*sin(t)-(y2-my)*cos(t);
   if(a2>a1)
     dep=(a2-a1)/4;
   else
    dep=(a1-a2)/4;
```

```
   bar3d(a1,b1,a2,b2,dep,1);
   setcolor(5);
   //draw();

}
 void draw()
{
   bar3d(x1,y1,x2,y2,depth,1);
}
void scale();
//these are left,top,right,bottom parameters for bar3d function
int maxx,maxy,midx,midy;
//function for scaling of a 3d object
void scale()
{
        int x,y,z,o,x1,x2,y1,y2;
        midx=200;
        midy=200;
        bar3d(midx+50,midy-100,midx+100,midy-50,20,0);
        printf("before scaling\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*100),midy-(y*50),20*z,1);
```

```
}

 void Line(constint,constint,constint,constint);
double face1[5][2] = {
      { 250, 125 },
      { 350, 125 },
      { 350, 225 },
      { 250, 225 },
      { 250, 125 }
           };

double face2[5][2] = {
      { 250+ORG, 125-ORG },
      { 350+ORG, 125-ORG },
      { 350+ORG, 225-ORG },
      { 250+ORG, 225-ORG },
      { 250+ORG, 125-ORG }
           };

double angle = 5.0 * M_PI / 180;
double midx1, midy1, midx2, midy2;

void rotate (void)
{
```

```
  int i;
  for (i=0; i<5; i++)
  {
double xnew, ynew;

xnew = midx1 + (face1[i][0] - midx1) * cos (angle) -
      (face1[i][1] - midy1) * sin (angle);
ynew = midy1 + (face1[i][0] - midx1) * sin (angle) +
 (face1[i][1] - midy1) * cos (angle);

face1[i][0] = xnew;
face1[i][1] = ynew;

xnew = midx2 + (face2[i][0] - midx2) * cos (angle) -
 (face2[i][1] - midy2) * sin (angle);
ynew = midy2 + (face2[i][0] - midx2) * sin (angle) +
 (face2[i][1] - midy2) * cos (angle);

face2[i][0] = xnew;
face2[i][1] = ynew;
  }

  cleardevice();
```

```
   for (i=0; i<4; i++)
   {
 setcolor(7);
 line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);
 setcolor(8);
 line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);
 setcolor(9);
 line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);
   }

   delay (125);
}


 void show_screen( );

 void apply_x_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);

 void draw_pyramid(constint [5][3]);
 void get_projected_point(int&,int&,int&);

 void Line(constint,constint,constint,constint);
```

```
int main( )
  {
     int driver=VGA;
     int mode=VGAHI;

     initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");

     show_screen( );

     int pyramid[5][3]={
            {280,220,40},     //  base front left
            {360,220,40},     //  base front right
            {360,220,-40},    //  base back right
            {280,220,-40},    //  base back left
            {320,100,0}       //  top
           };

    setcolor(15);
   draw_pyramid(pyramid);

     setcolor(15);
     settextstyle(0,0,1);
   outtextxy(50,415,"*** Press any key to see the 3D Shearing along x-axis.");
```

```
    apply_x_shearing(pyramid,0.4,0.3);

    getch( );

    setcolor(10);
   draw_pyramid(pyramid);

    getch( );
    return 0;
  }


void apply_x_shearing(int edge_points[5][3],constfloat a,constfloat b)
  {
    for(int count=0;count<5;count++)
   {
      float matrix_a[4]={edge_points[count][0],edge_points[count][1],
                  edge_points[count][2],1};
      float matrix_b[4][4]={
            { 1,a,b,0 } ,
            { 0,1,0,0 } ,
            { 0,0,1,0 } ,
            { 0,0,0,1 }
```

```
            };

        float matrix_c[4]={0};

        multiply_matrices(matrix_a,matrix_b,matrix_c);

        edge_points[count][0]=(int)(matrix_c[0]+0.5);
        edge_points[count][1]=(int)(matrix_c[1]+0.5);
        edge_points[count][2]=(int)(matrix_c[2]+0.5);
      }
    }

void multiply_matrices(constfloat matrix_1[4],
              constfloat matrix_2[4][4],float matrix_3[4])
    {
      for(int count_1=0;count_1<4;count_1++)
      {
        for(int count_2=0;count_2<4;count_2++)
       matrix_3[count_1]+=
            (matrix_1[count_2]*matrix_2[count_2][count_1]);
      }
    }

void draw_pyramid(constint points[5][3])
```

```
{
  int edge_points[5][3];

  for(int i=0;i<5;i++)
  {
    edge_points[i][0]=points[i][0];
    edge_points[i][1]=points[i][1];
    edge_points[i][2]=points[i][2];

    get_projected_point(edge_points[i][0],
            edge_points[i][1],edge_points[i][2]);
  }

  Line(edge_points[0][0],edge_points[0][1],
              edge_points[1][0],edge_points[1][1]);
  Line(edge_points[1][0],edge_points[1][1],
              edge_points[2][0],edge_points[2][1]);
  Line(edge_points[2][0],edge_points[2][1],
              edge_points[3][0],edge_points[3][1]);
  Line(edge_points[3][0],edge_points[3][1],
              edge_points[0][0],edge_points[0][1]);

  Line(edge_points[0][0],edge_points[0][1],
              edge_points[4][0],edge_points[4][1]);
```

```
    Line(edge_points[1][0],edge_points[1][1],
                edge_points[4][0],edge_points[4][1]);
    Line(edge_points[2][0],edge_points[2][1],
                edge_points[4][0],edge_points[4][1]);
    Line(edge_points[3][0],edge_points[3][1],
                edge_points[4][0],edge_points[4][1]);
  }

void get_projected_point(int& x,int& y,int& z)
  {
    float fcos0=(f*cos(projection_angle*(M_PI/180)));
    float fsin0=(f*sin(projection_angle*(M_PI/180)));

    float Par_v[4][4]={
          {1,0,0,0},
          {0,1,0,0},
          {fcos0,fsin0,0,0},
          {0,0,0,1}
        };

    float xy[4]={x,y,z,1};
    float new_xy[4]={0};

    multiply_matrices(xy,Par_v,new_xy);
```

```
    x=(int)(new_xy[0]+0.5);
    y=(int)(new_xy[1]+0.5);
    z=(int)(new_xy[2]+0.5);
  }

void Line(constint x_1,constint y_1,constint x_2,constint y_2)
  {
    int color=getcolor( );

    int x1=x_1;
    int y1=y_1;

    int x2=x_2;
    int y2=y_2;

    if(x_1>x_2)
    {
      x1=x_2;
      y1=y_2;

      x2=x_1;
      y2=y_1;
    }
```

```
int dx=abs(x2-x1);
int dy=abs(y2-y1);
int inc_dec=((y2>=y1)?1:-1);

if(dx>dy)
{
  int two_dy=(2*dy);
  int two_dy_dx=(2*(dy-dx));
  int p=((2*dy)-dx);

  int x=x1;
  int y=y1;

  putpixel(x,y,color);

  while(x<x2)
  {
    x++;

    if(p<0)
      p+=two_dy;

    else
```

```
        {
        y+=inc_dec;
        p+=two_dy_dx;
        }

     putpixel(x,y,color);
   }
 }

 else
 {
    int two_dx=(2*dx);
    int two_dx_dy=(2*(dx-dy));
    int p=((2*dx)-dy);

    int x=x1;
    int y=y1;

    putpixel(x,y,color);

    while(y!=y2)
   {
     y+=inc_dec;
```

```
        if(p<0)
          p+=two_dx;

        else
          {
         x++;
         p+=two_dx_dy;
          }

        putpixel(x,y,color);
      }
     }
   }

void show_screen( )
  {
     setfillstyle(1,1);
    bar(210,26,420,38);

     settextstyle(0,0,1);
    setcolor(15);

outtextxy(5,5,"**********************************************************************
***********");
```

```
    outtextxy(5,17,"*-
*************************************************************************-*");
    outtextxy(5,29,"*----------------------                    ----------------------*");
    outtextxy(5,41,"*-
*************************************************************************-*");
    outtextxy(5,53,"*-
*************************************************************************-*");

   setcolor(11);
    outtextxy(218,29,"3D Shearing along x-axis");

   setcolor(15);

    for(int count=0;count<=30;count++)
      outtextxy(5,(65+(count*12)),"*-*                                                    *-*");

    outtextxy(5,438,"*-
*************************************************************************-*");
    outtextxy(5,450,"*------------------------                  ------------------------*");

outtextxy(5,462,"*************************************************************
**************");

   setcolor(12);
    outtextxy(229,450,"Press any Key to exit.");
  }
```

```
int main () {
 char choice;
 printf("Enter 1 for translation,2 for reflection,3 for rotation,4 for scaling,5 for shearing along x axis,6 for shearing along y axis.\n");
 scanf("%c", &choice)
 switch(choice) {
 case '1' :int ch;

        int gd=DETECT,gm;

        detectgraph(&gd,&gm);

        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

        trans();
break;
case '2' :
 int gd = DETECT, gm;

   midx1 = (face1[0][0] + face1[1][0]) / 2.0;
   midy1 = (face1[1][1] + face1[2][1]) / 2.0;
   midx2 = (face2[0][0] + face2[1][0]) / 2.0;
   midy2 = (face2[1][1] + face2[2][1]) / 2.0;
```

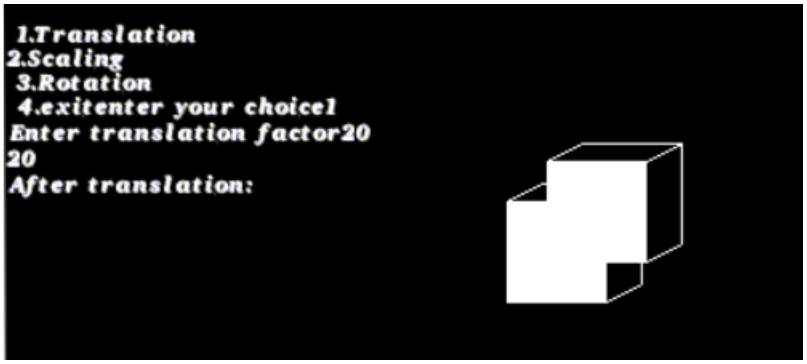
```
   initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");

   while (!kbhit())
 rotate();

   closegraph();
break;
 case '3' :
int gd=DETECT,gm,c;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   printf("\n3D Transformation Rotating\n\n");
   printf("\nEnter 1st top value(x1,y1):");
   scanf("%d%d",&x1,&y1);
   printf("Enter right bottom value(x2,y2):");
   scanf("%d%d",&x2,&y2);
   depth=(x2-x1)/4;
   mx=(x1+x2)/2;
   my=(y1+y2)/2;
   draw();
   getch();
   cleardevice();
   rotate();
   getch();
 break;
```

```
case '4' : int ch;
          int gd=DETECT,gm;
          detectgraph(&gd,&gm);
          initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
          scale();
break;
case '5':int driver=VGA;
     int mode=VGAHI;

     initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");

     show_screen( );

     int pyramid[5][3]={
            {280,220,40},     //  base front left
            {360,220,40},     //  base front right
            {360,220,-40},    //  base back right
            {280,220,-40},    //  base back left
            {320,100,0}       //  top
           };

    setcolor(15);
   draw_pyramid(pyramid);
```

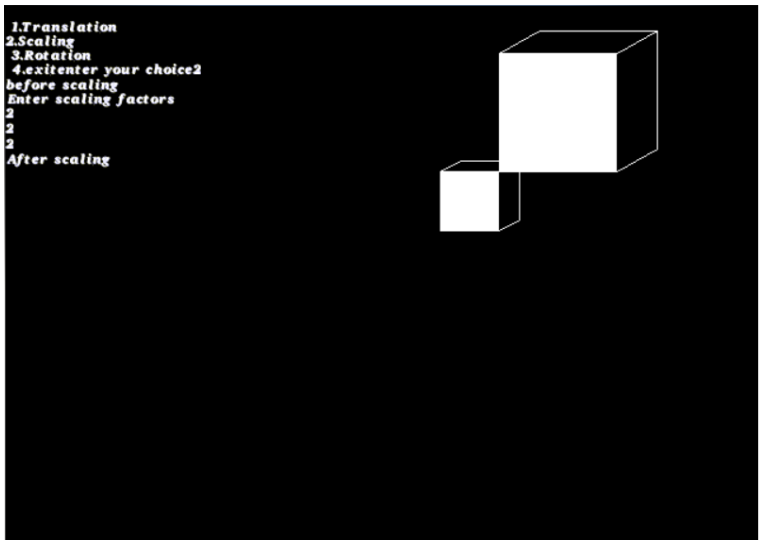
```
      setcolor(15);
      settextstyle(0,0,1);
     outtextxy(50,415,"*** Press any key to see the 3D Shearing along x-axis.");

      apply_x_shearing(pyramid,0.4,0.3);

      getch( );

      setcolor(10);
     draw_pyramid(pyramid);

      getch( );
break;
 case '6':int driver=VGA;
      int mode=VGAHI;

      initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");

      show_screen( );

      int pyramid[5][3]={
            {270,300,50},     //  base front left
            {370,300,50},     //  base front right
            {370,300,-50},    //  base back right
```

```
            {270,300,-50},     //  base back left
            {320,150,0}       //  top
           };

     setcolor(15);
    draw_pyramid(pyramid);

      setcolor(15);
      settextstyle(0,0,1);
    outtextxy(50,415,"*** Press any key to see the 3D Shearing along y-axis.");

      apply_y_shearing(pyramid,0.5,0.1);

      getch( );

      setcolor(10);
    draw_pyramid(pyramid);

      getch( );
  default :
  printf("Wrong Choice.Try Again.\n" );
  }
  return 0;
 }
```

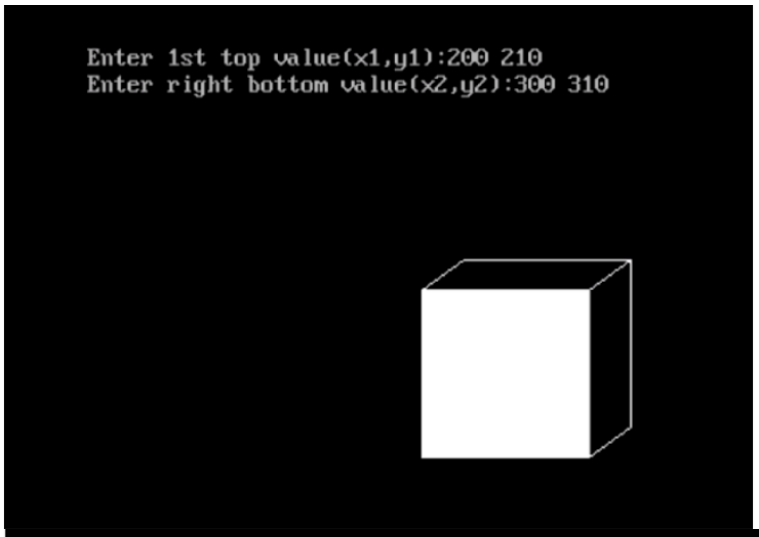
TRANSLATION:

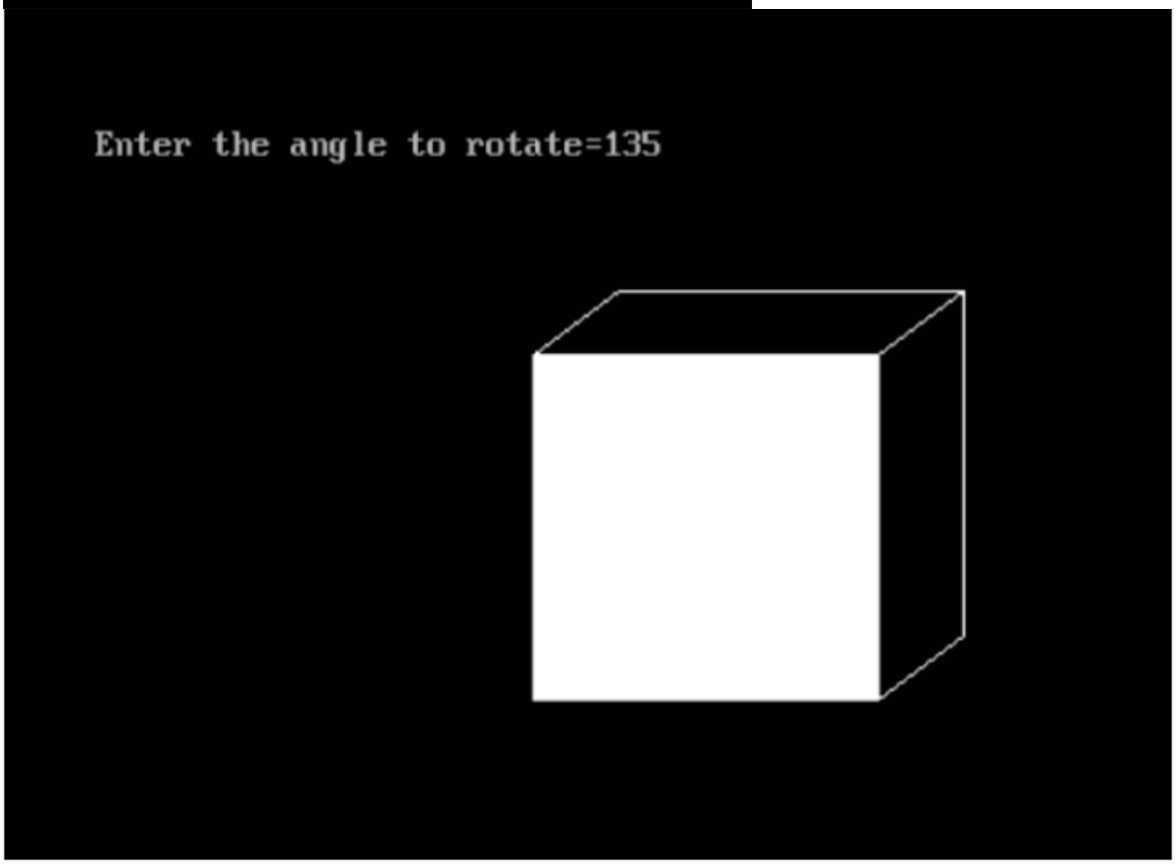


SCALING:

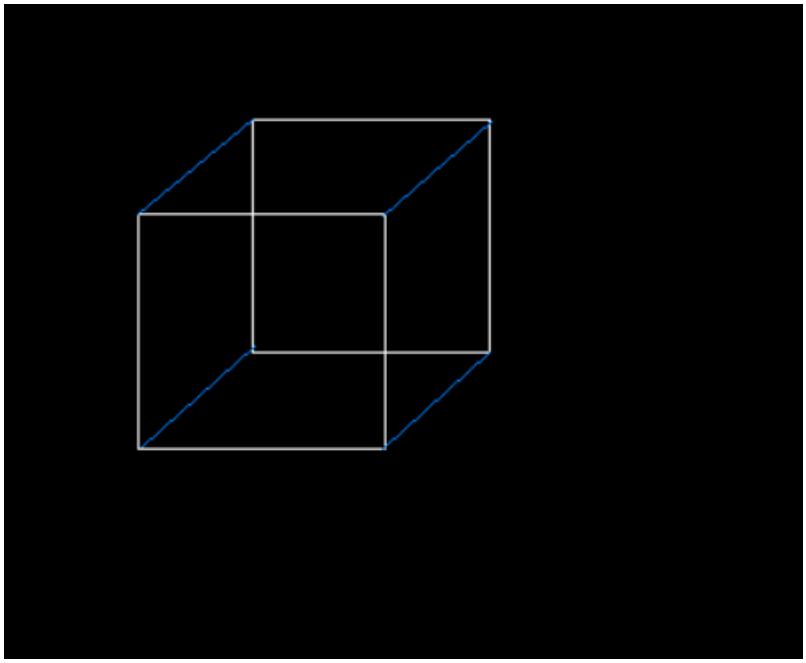


ROTATION:

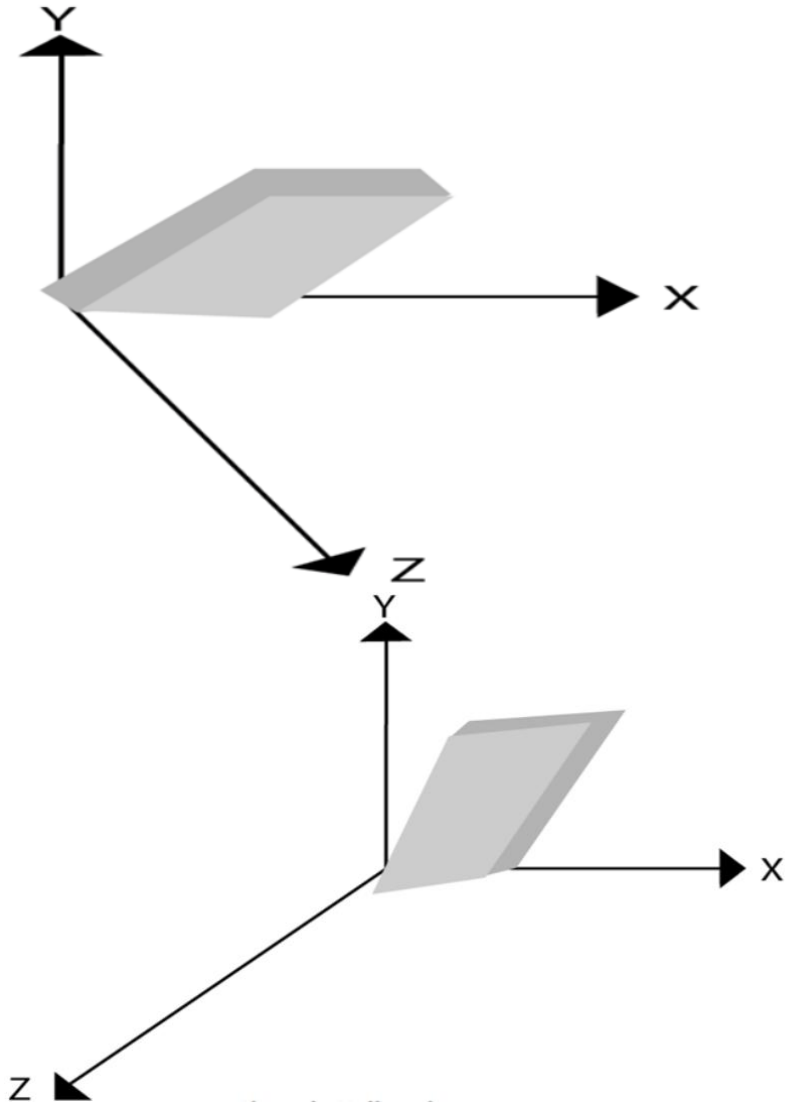

Enter 1st top value(x1,y1):200 210
Enter right bottom value(x2,y2):300 310


Enter the angle to rotate=135

REFLECTION:



SHEARING:

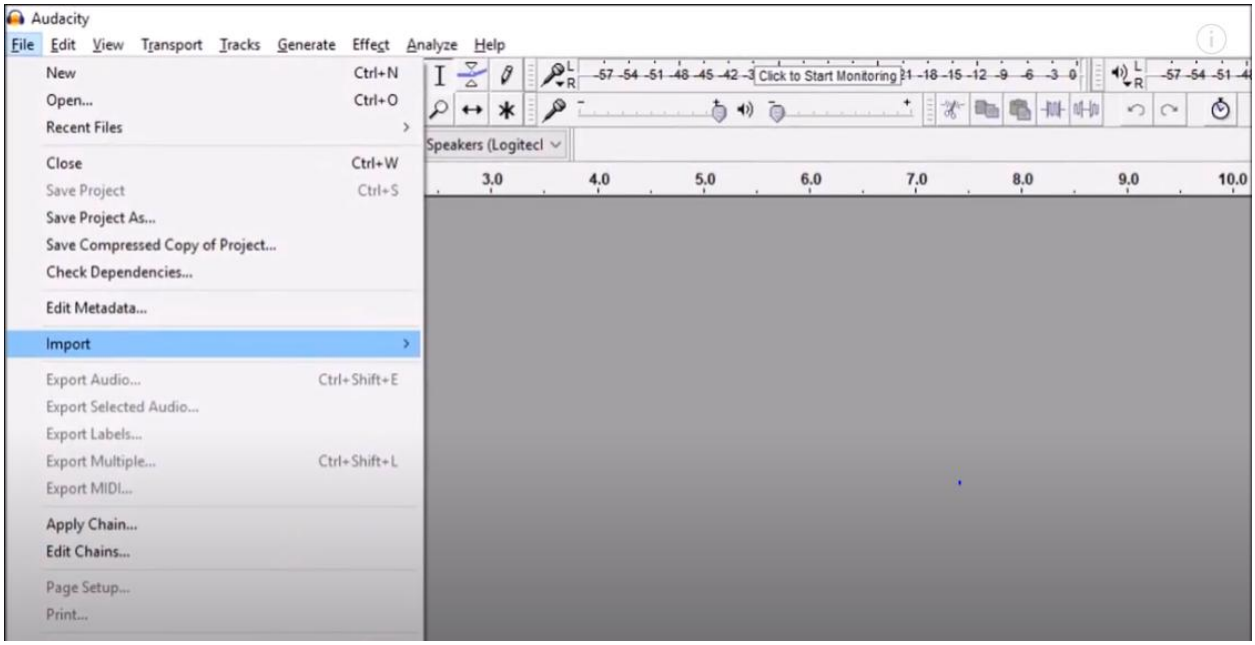


Lab 8:Audio Editing

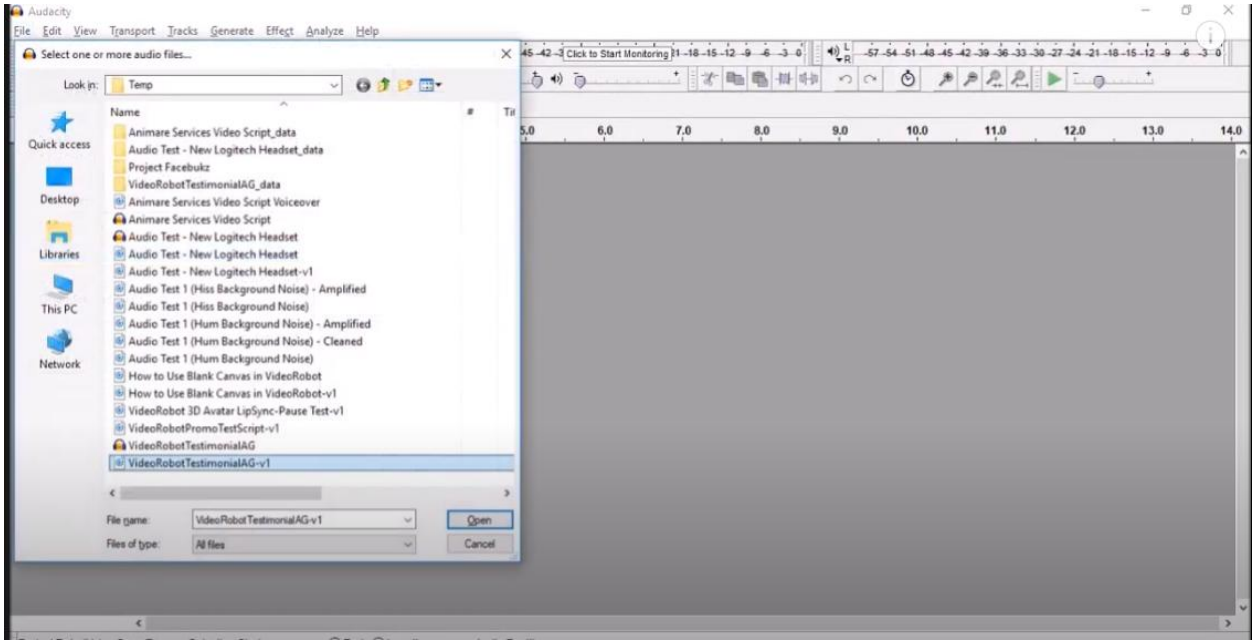
Drive Link: https://drive.google.com/drive/folders/10abxunpu08QHtXNJNo_4haiAAFIzsMr6?usp=sharing

Use a audio processing software and perform the audio editing tasks–

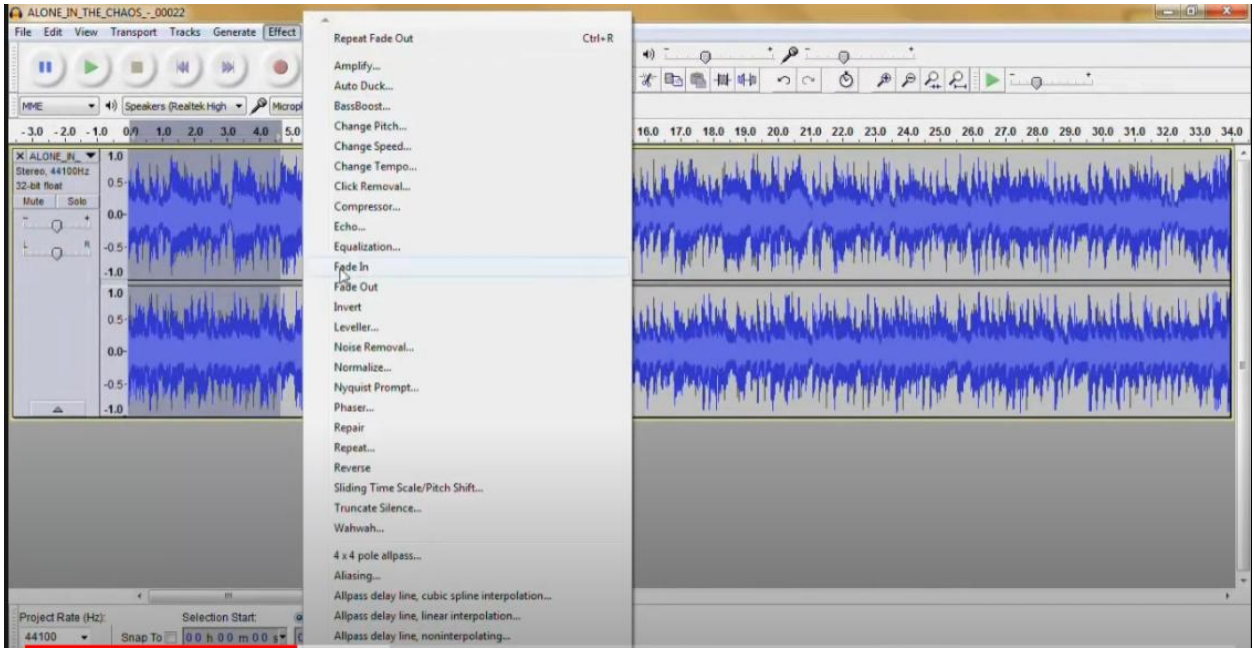
Import audio

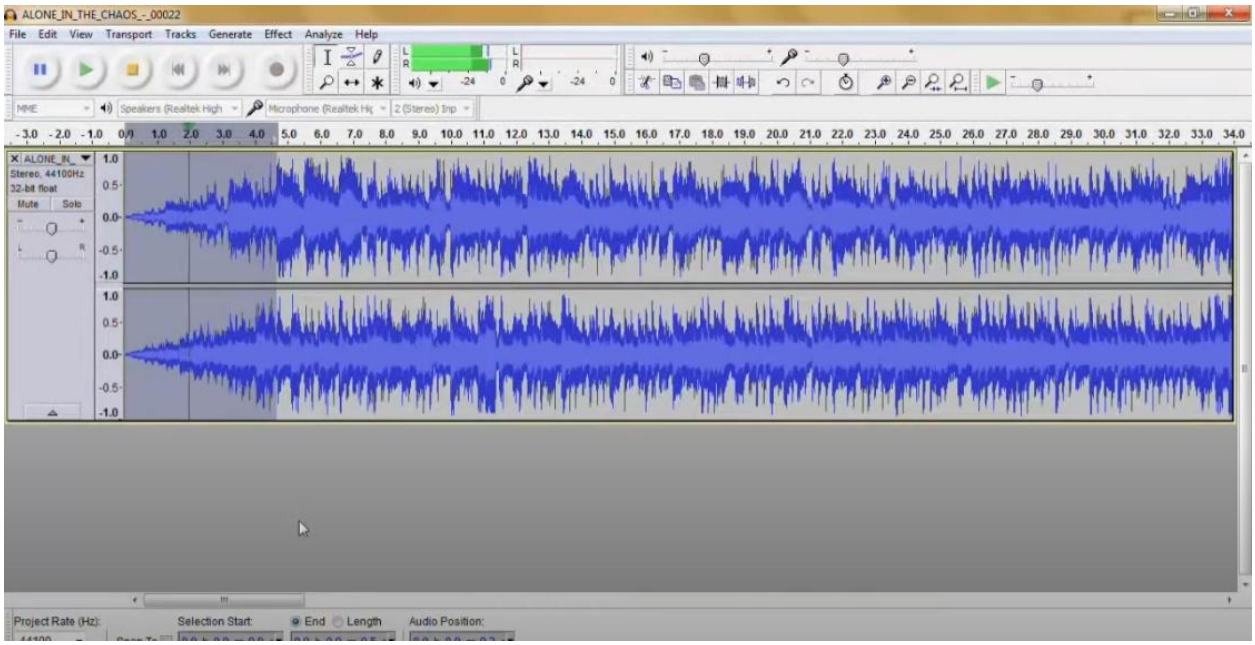


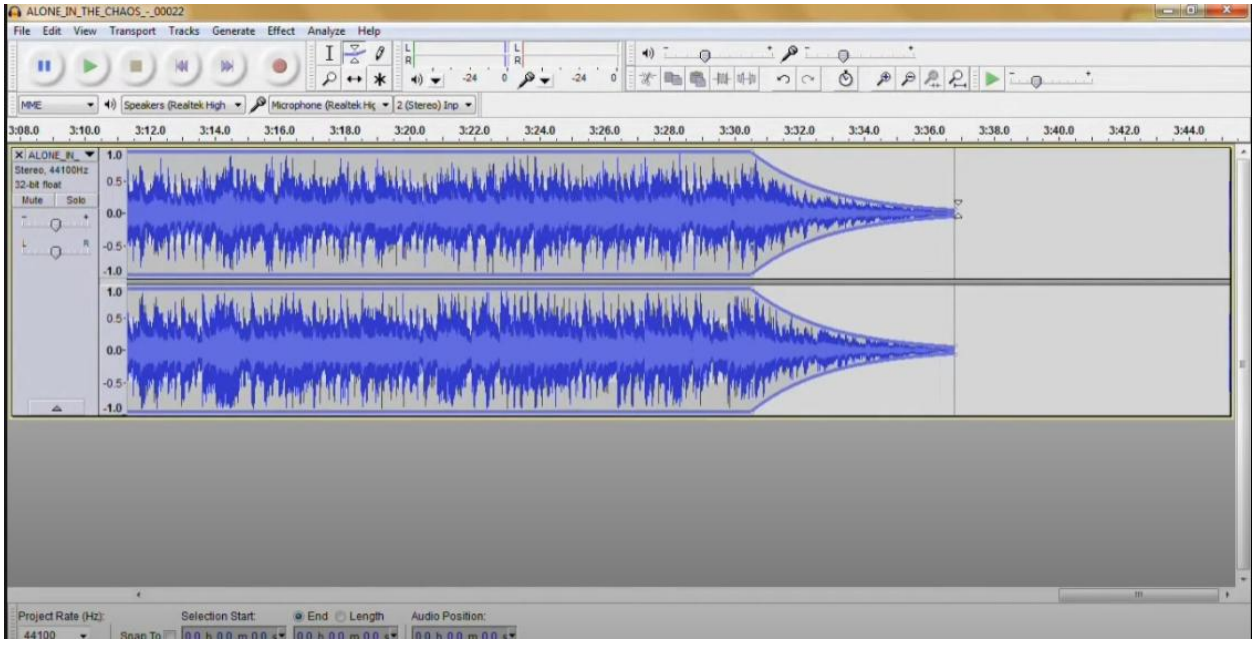
Select and edit the sound



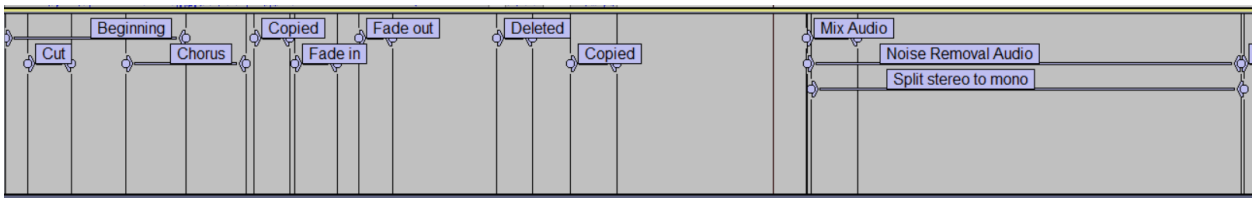
Create fade-in fade-out effects



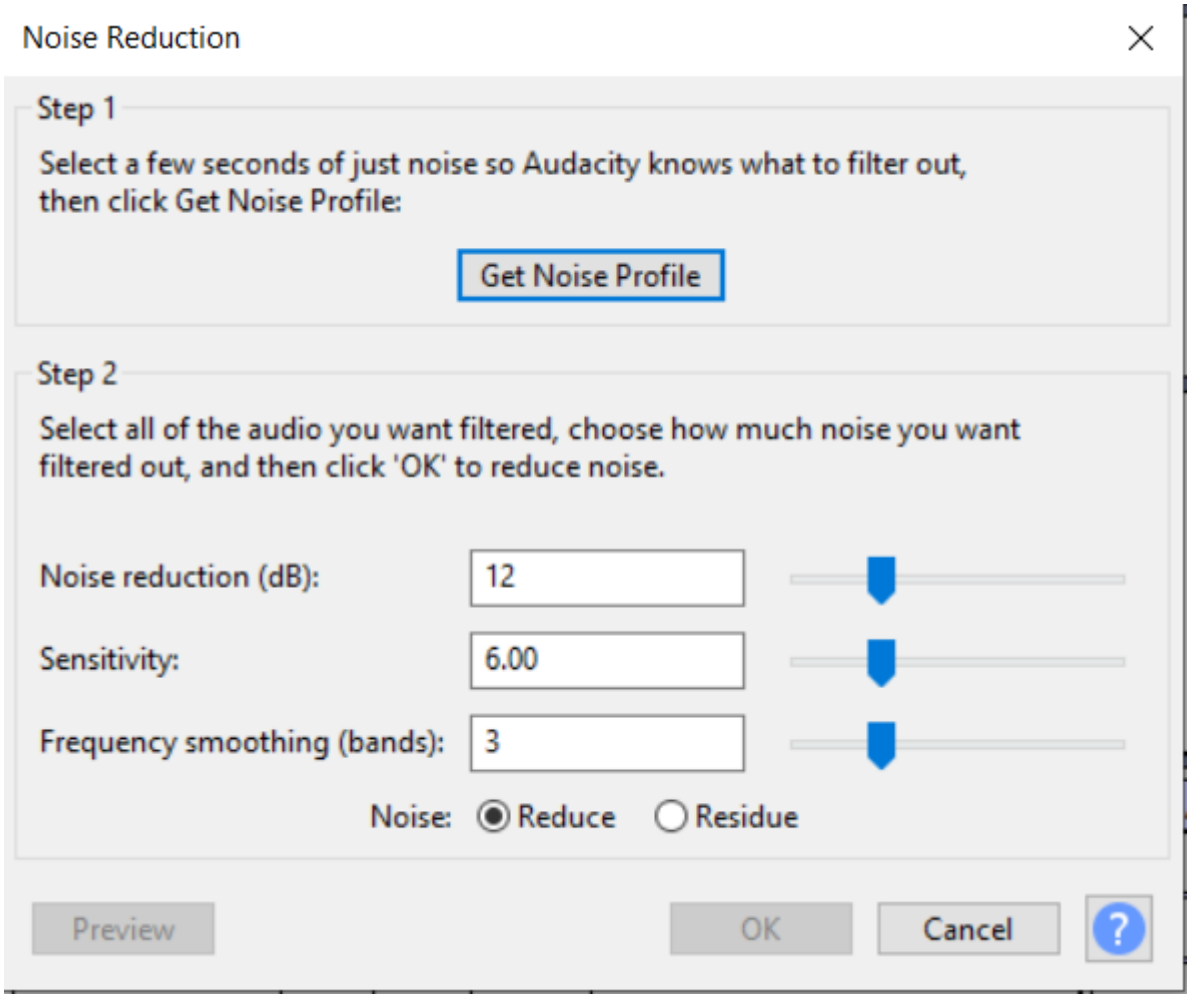


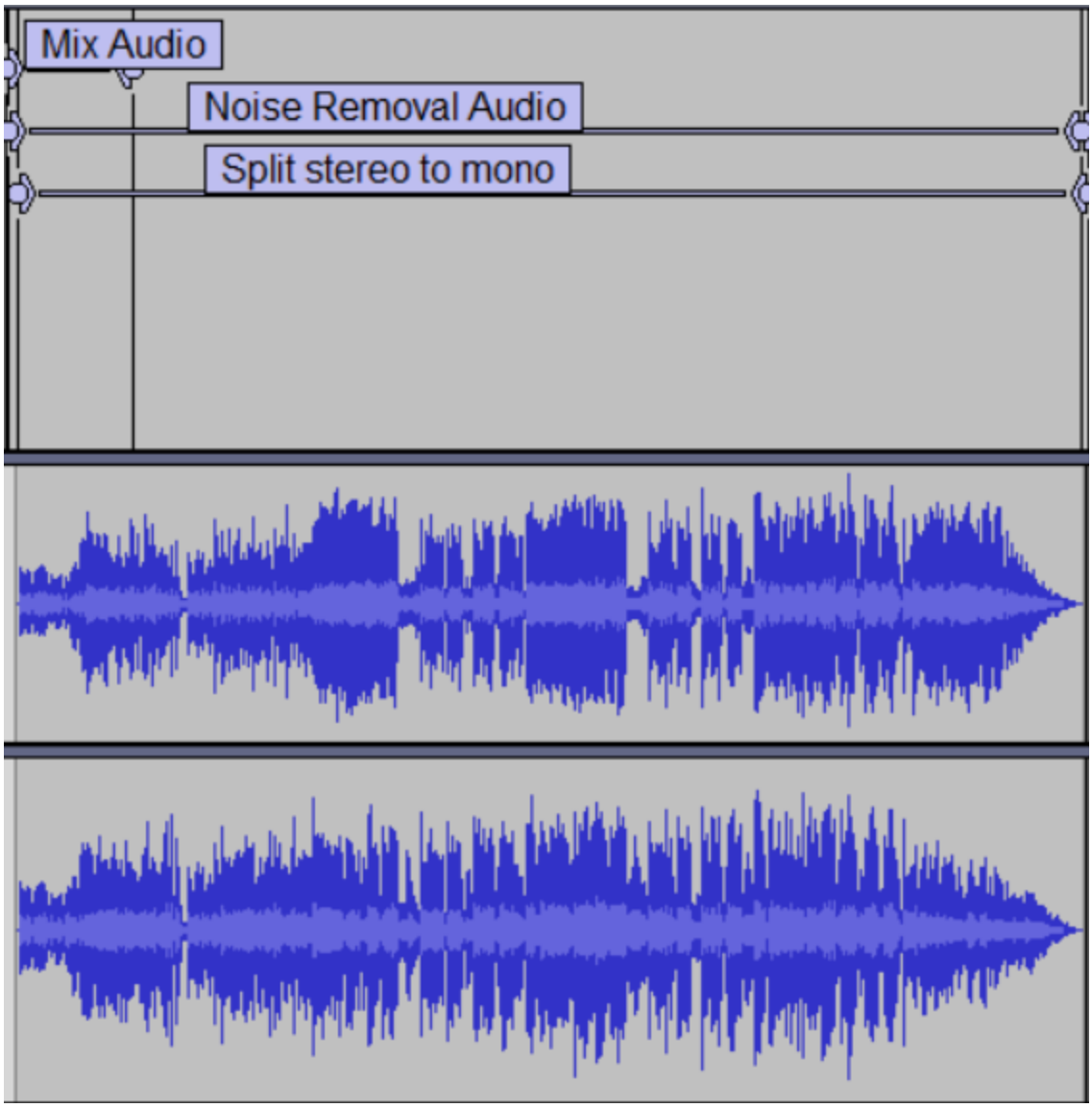


, Label audio segments,

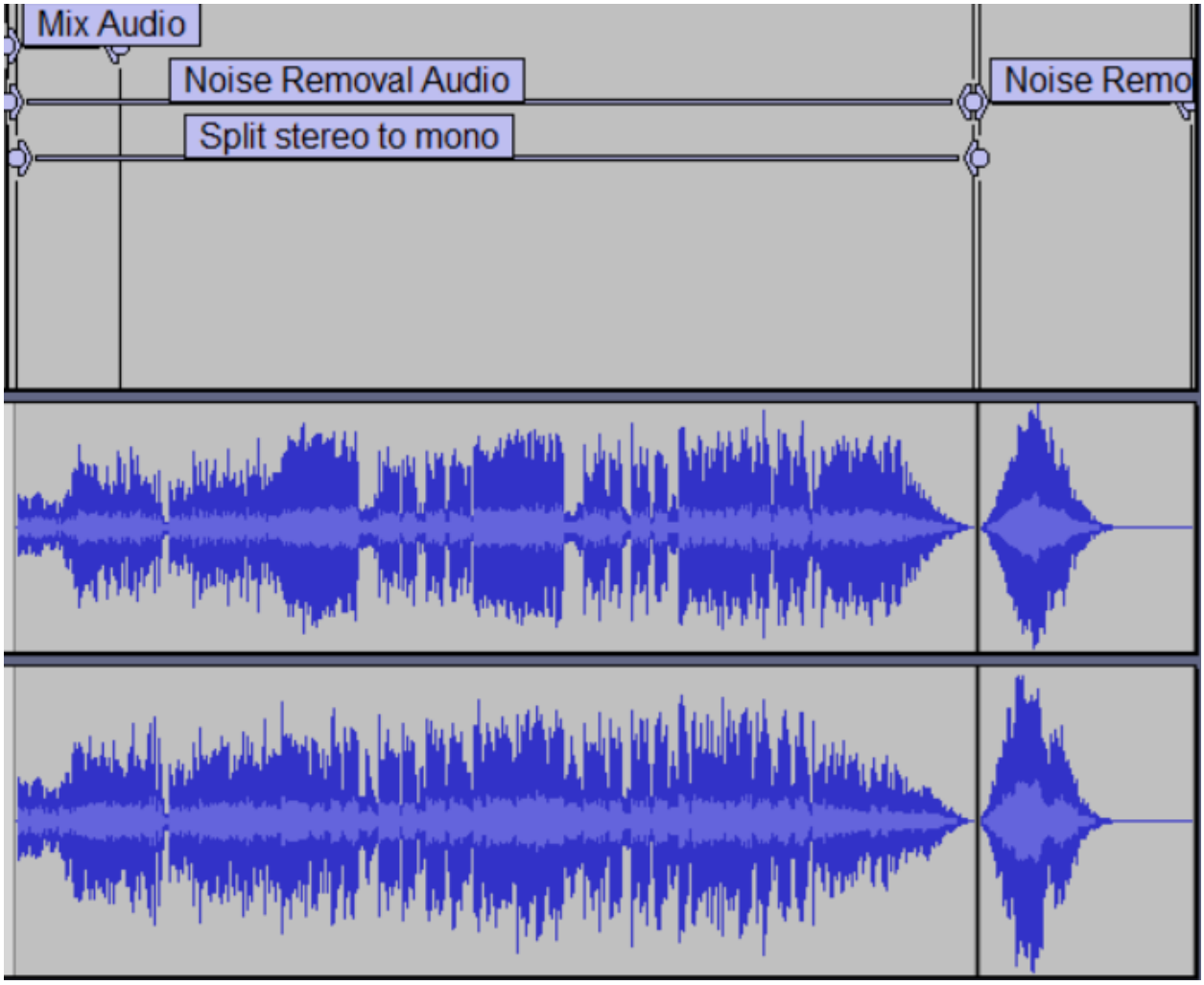


Use noise remove filter

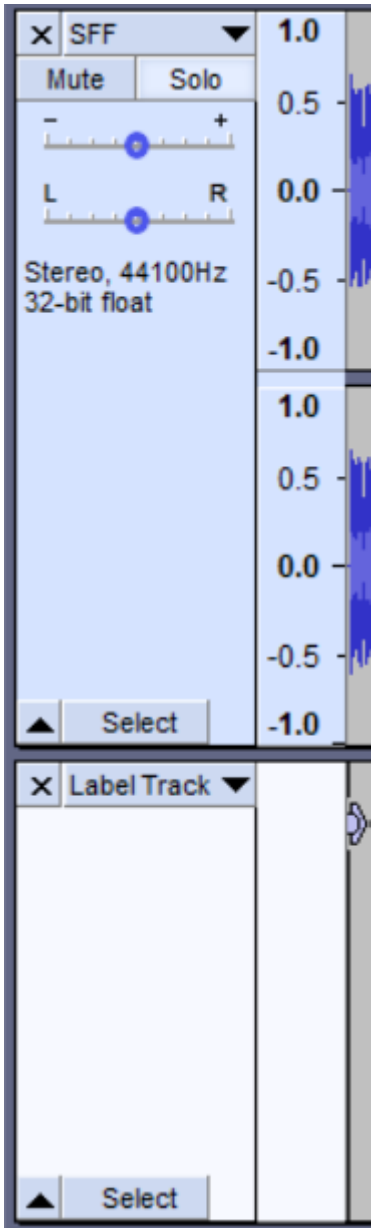

Noise Reduction
Step 1
Select a few seconds of just noise so Audacity knows what to filter out, then click Get Noise Profile:
Get Noise Profile
Step 2
Select all of the audio you want filtered, choose how much noise you want filtered out, and then click 'OK' to reduce noise.
Noise reduction (dB):
12
Sensitivity:
6.00
Frequency smoothing (bands):
3
Noise:
Reduce
Residue
Preview
OK
Cancel

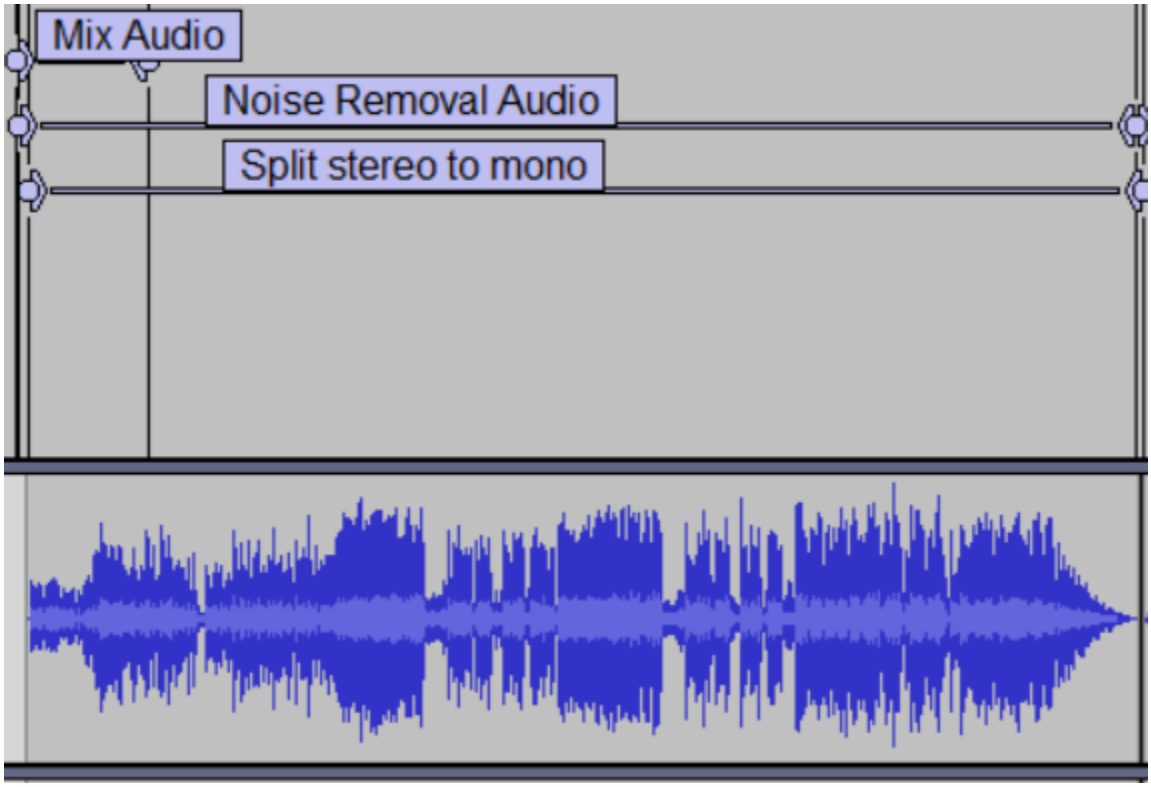

Mix Audio
Noise Removal Audio
Split stereo to mono

, Mix audio

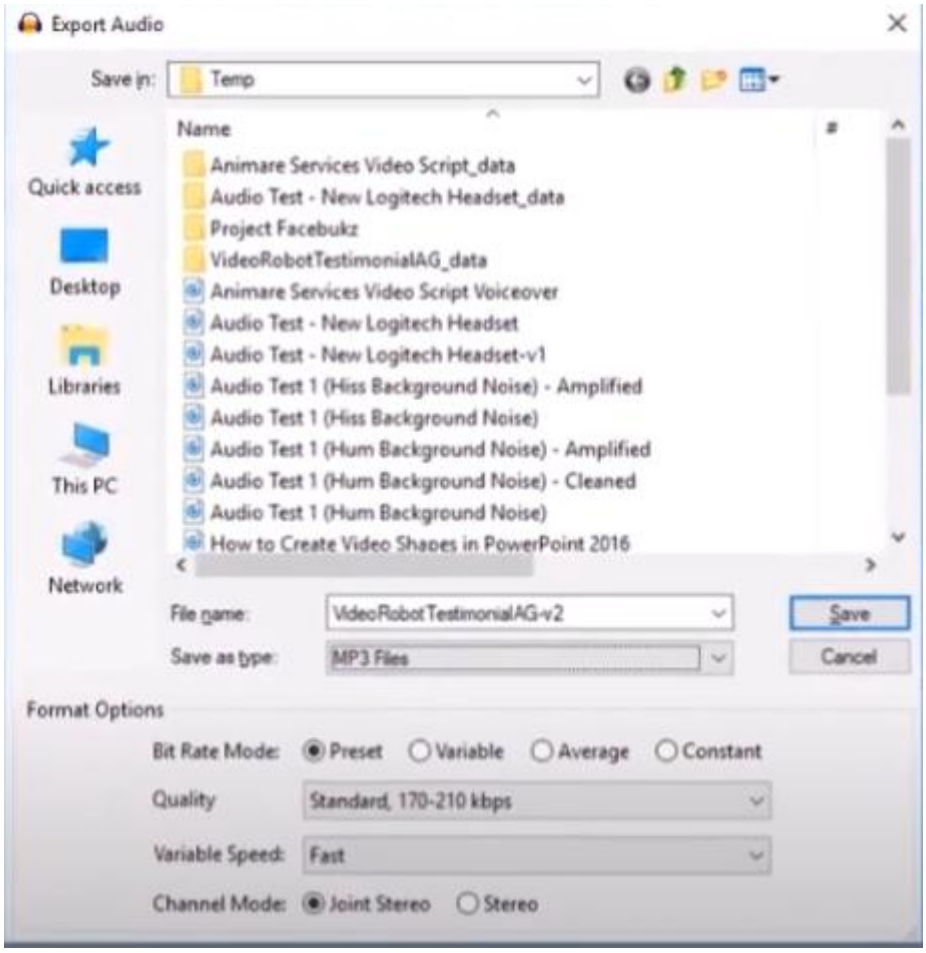


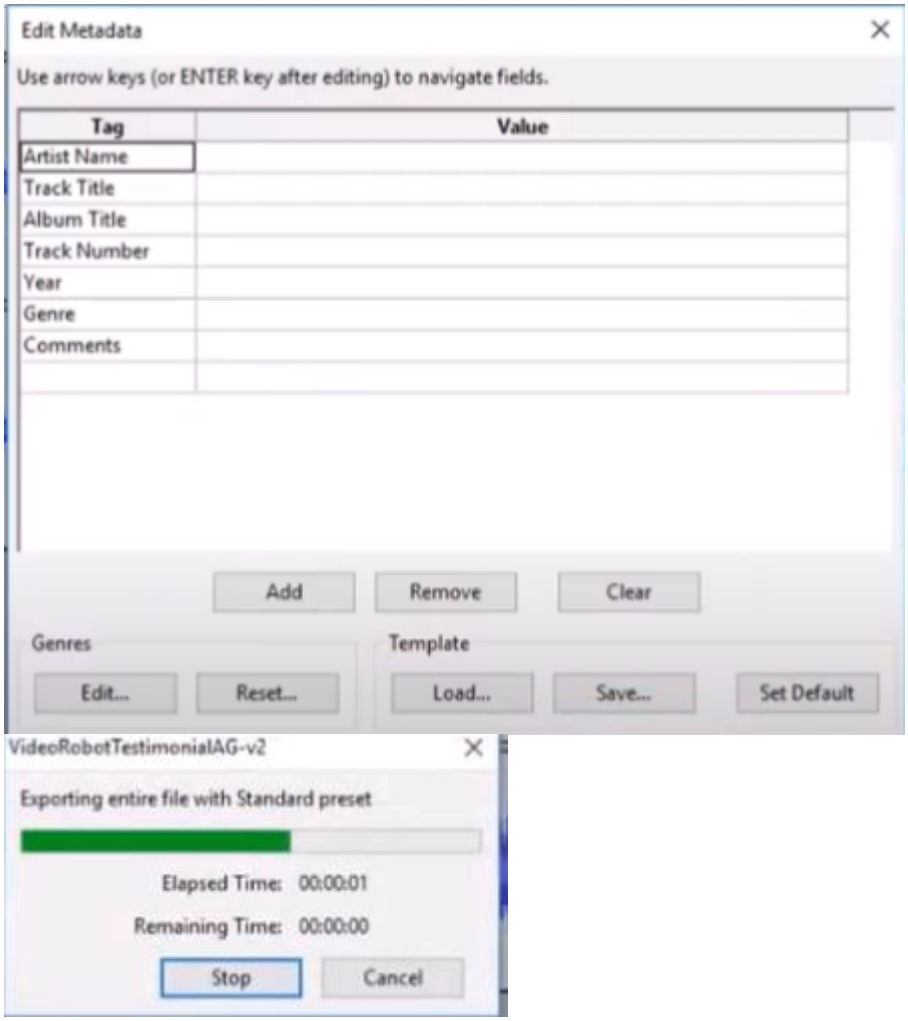
, Change stereo to mono tracks


SFF
Mute
Solo
L
R
Stereo, 44100Hz
32-bit float
Select
1.0
0.5
0.0
-0.5
-1.0
Label Track
Select


Mix Audio
Noise Removal Audio
Split stereo to mono

Export audio to different format and save.




Edit Metadata
Use arrow keys (or ENTER key after editing) to navigate fields.
Tag
Value
Artist Name
Track Title
Album Title
Track Number
Year
Genre
Comments
Add
Remove
Clear
Genres
Edit...
Reset...
Template
Load...
Save...
Set Default
VideoRobotTestimonialAG-v2
Exporting entire file with Standard preset
Elapsed Time: 00:00:01
Remaining Time: 00:00:00
Stop
Cancel

Lab 9:Video Editing

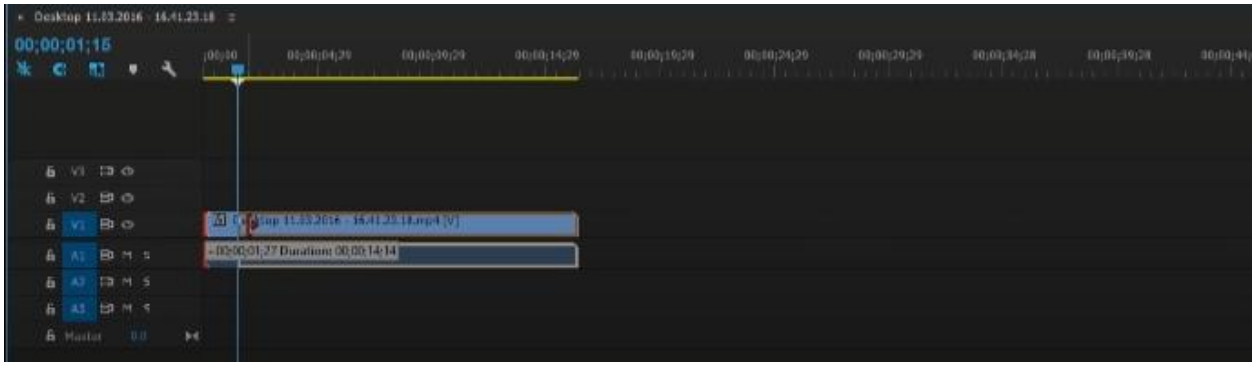
Use a video processing Software to perform – Trim video clips, crop video, rotate video, join video, add subtitles, and edit video dimension, bit rate, 3 hoursframe rate, sample rate, channel on a video.

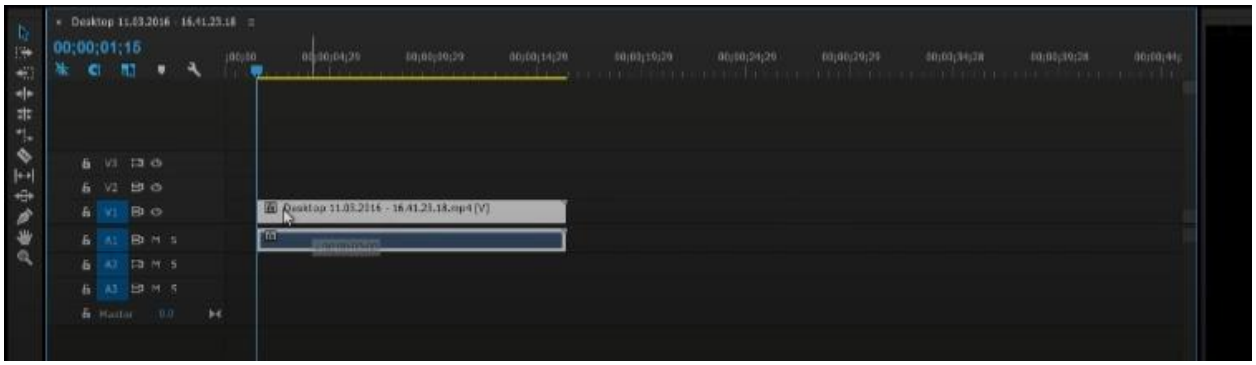
Premiere Pro CC

Gdrive Link: https://drive.google.com/file/d/1M_jqFVGbfmOTnn0Ga1wL3Nne7JsGrxqI/view?usp=sharing

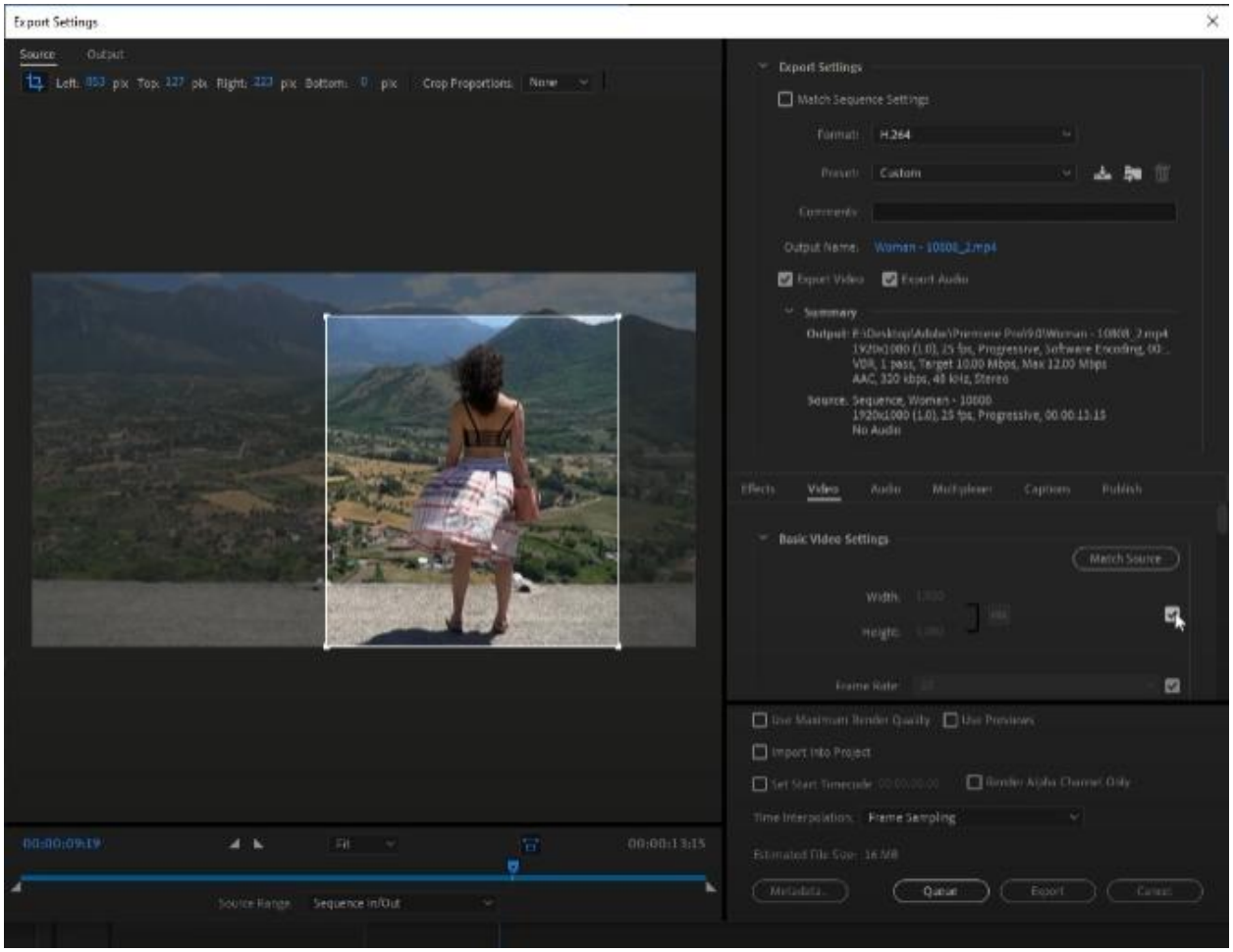
Video Link: https://www.youtube.com/watch?v=WBwvKGc9ySc

Trim video clips

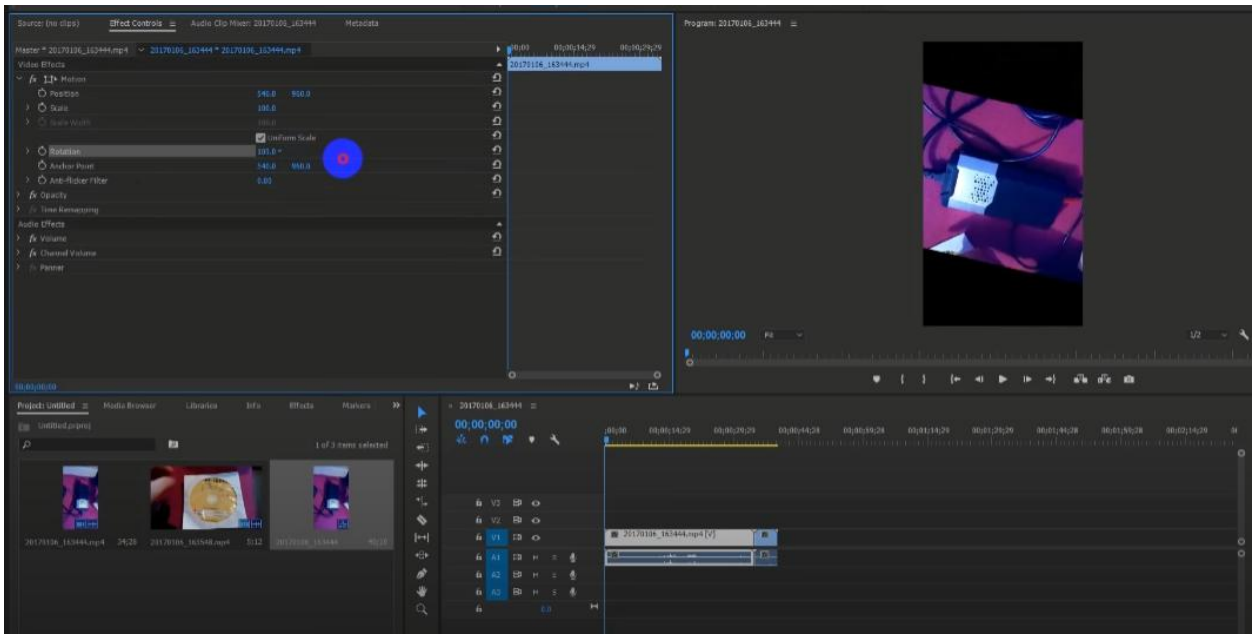




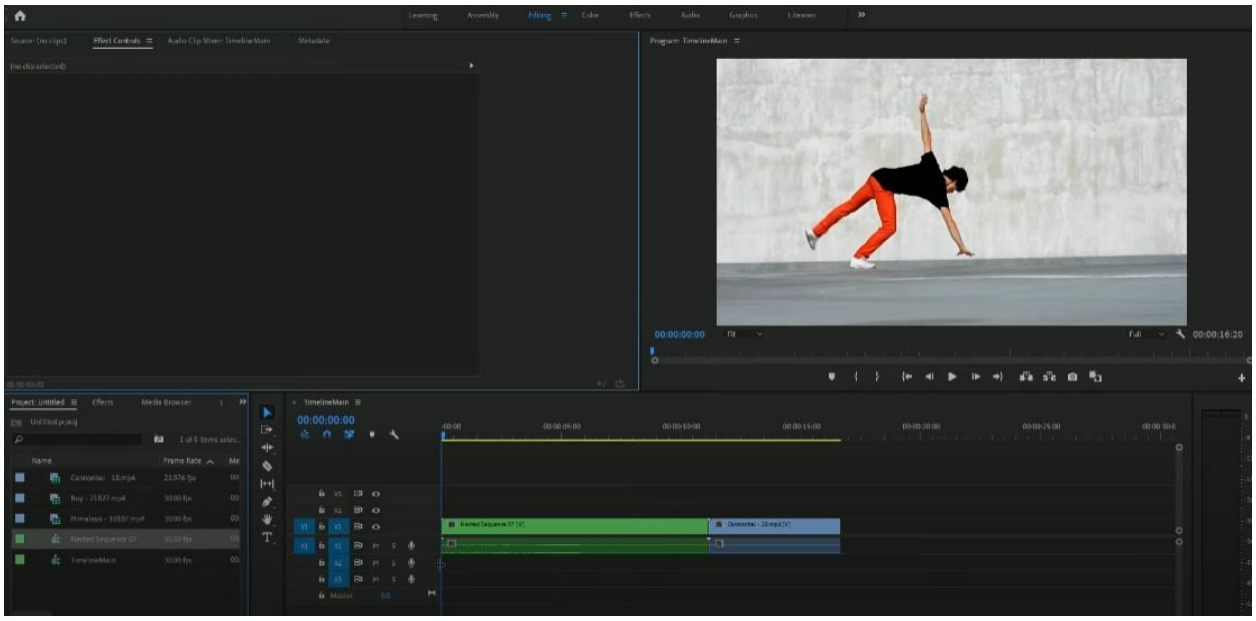
crop video



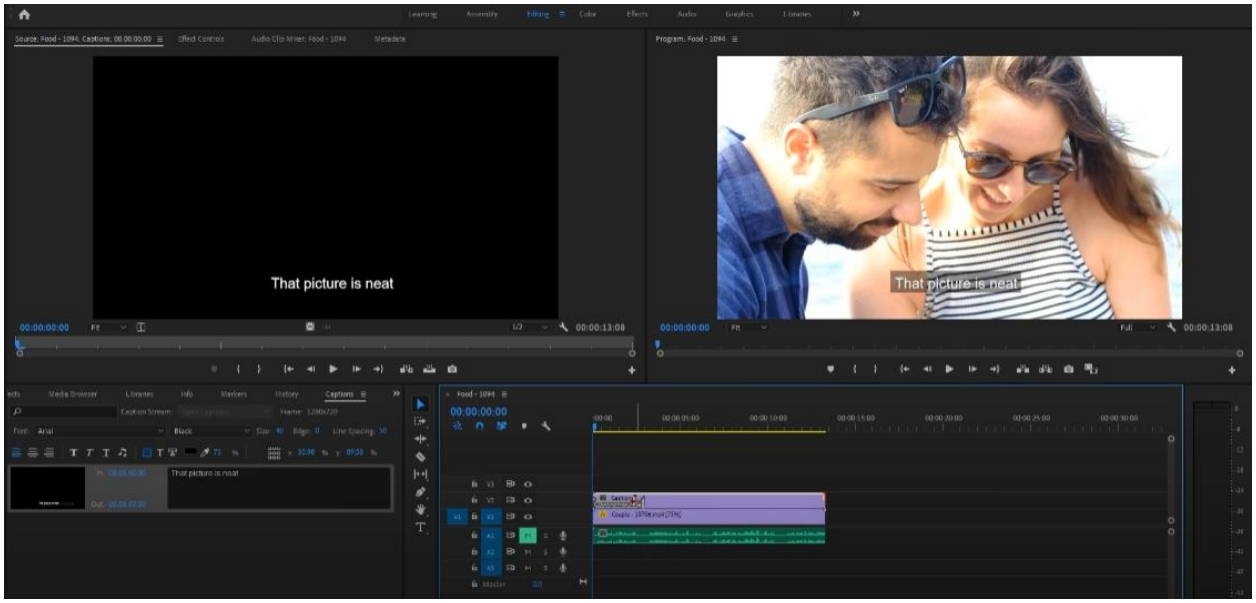
rotate video



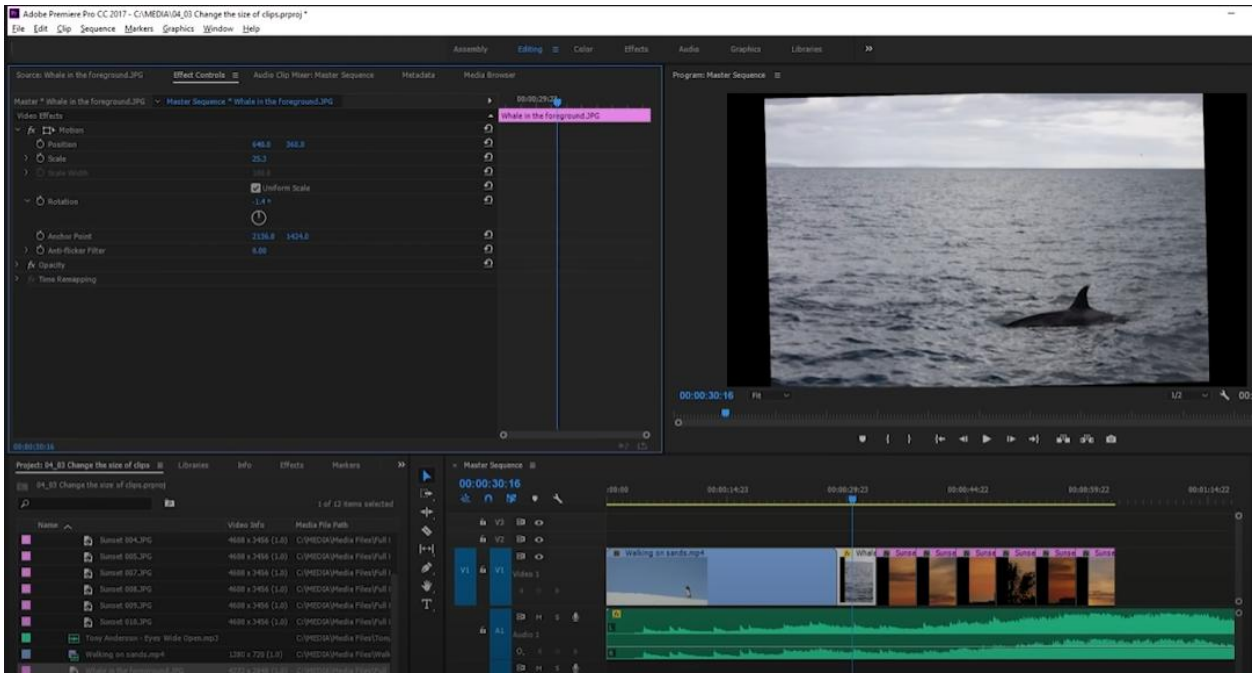
join video

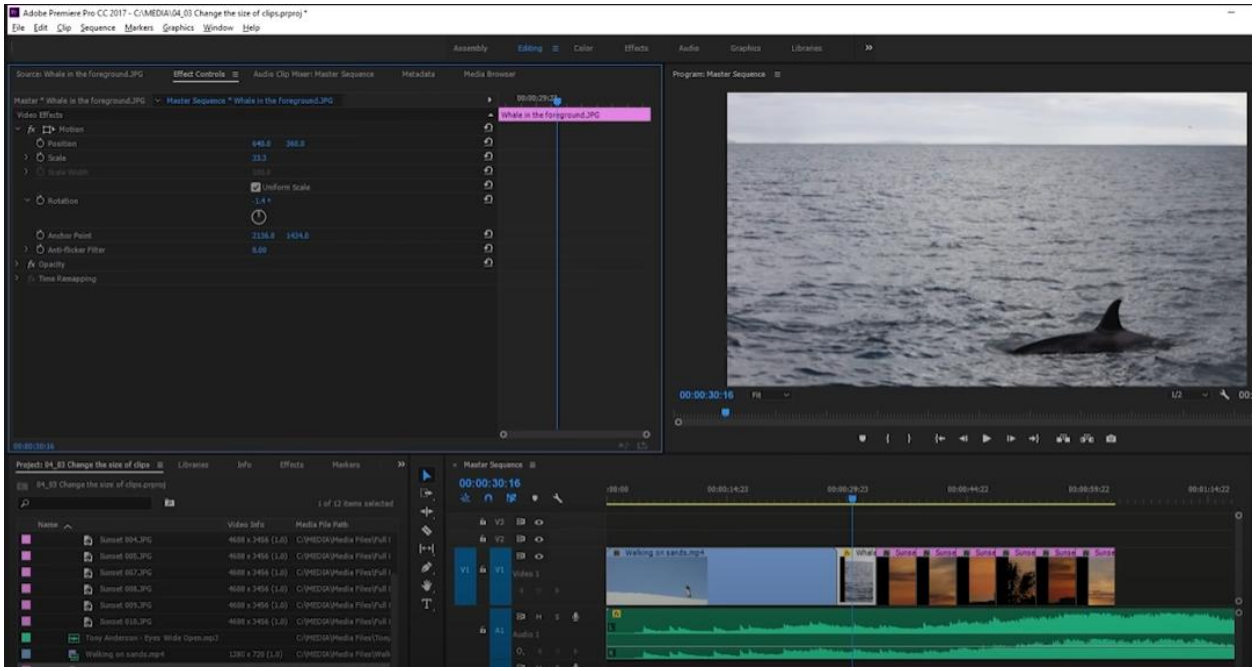


add subtitles

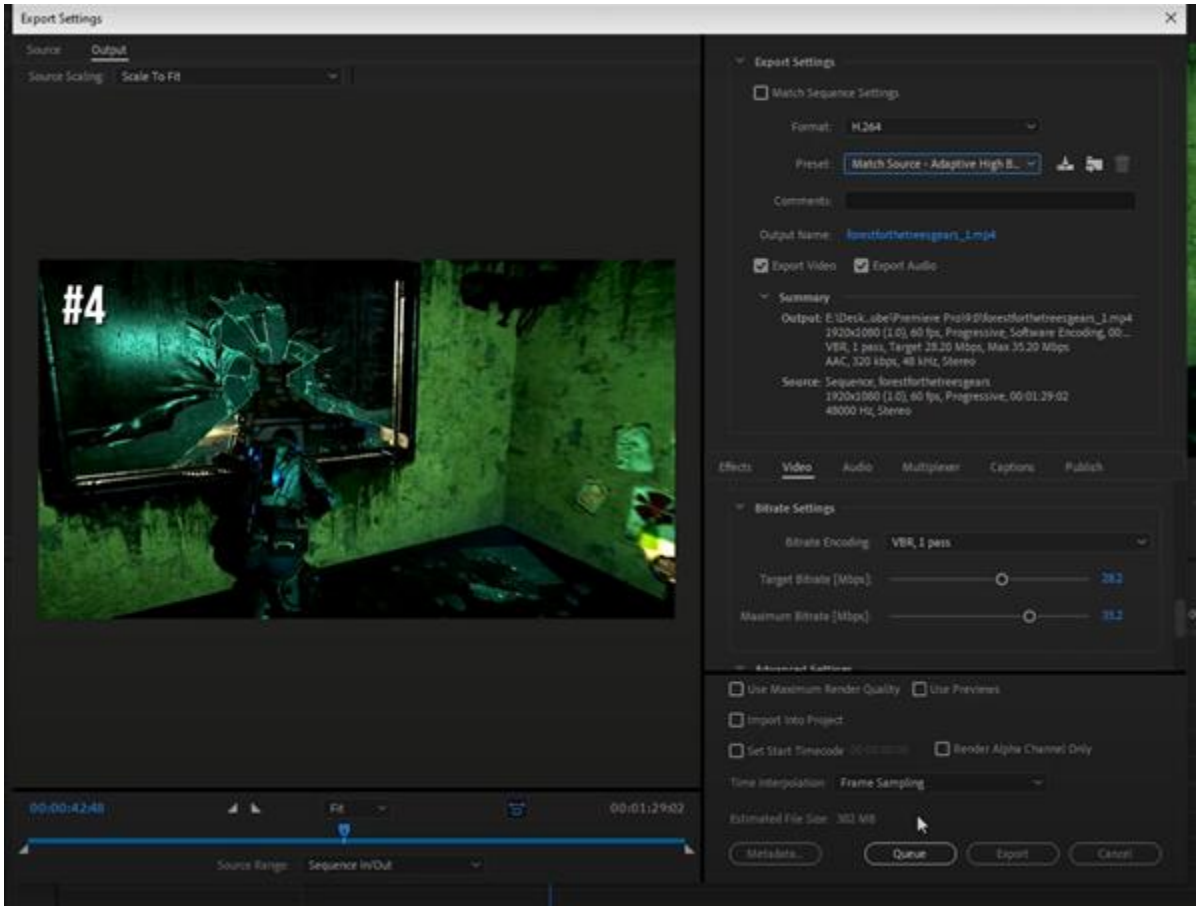


edit video dimension

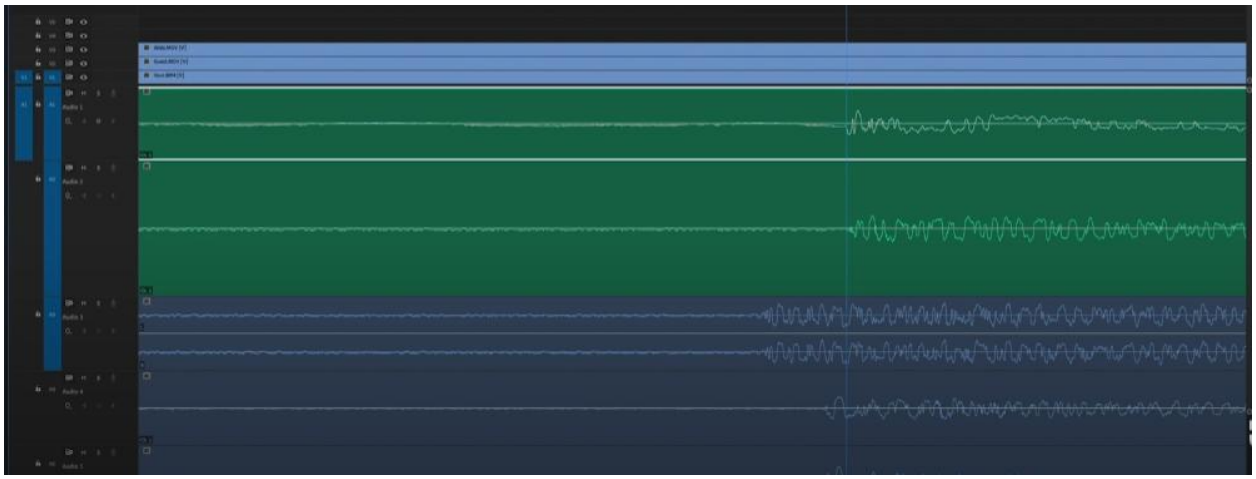




bit rate


Export Settings
Source
Output
Source Scaling: Scale To Fit
#4
Export Settings
Match Sequence Settings
Format: H.264
Preset: Match Source - Adaptive High B...
Comments:
Output Name: forestforthetreesgears_1.mp4
Export Video
Export Audio
Summary
Output: E:\Desk..obe\Premiere Pro\9.0\forestforthetreesgears_1.mp4
1920x1080 (1.0), 60 fps, Progressive, Software Encoding, 00:...
VBR, 1 pass, Target 28.20 Mbps, Max 35.20 Mbps
AAC, 320 kbps, 48 kHz, Stereo
Source: Sequence, forestforthetreesgears
1920x1080 (1.0), 60 fps, Progressive, 00:01:29:02
48000 Hz, Stereo
Effects
Video
Audio
Multiplexer
Captions
Publish
Bitrate Settings
Bitrate Encoding: VBR, 1 pass
Target Bitrate [Mbps]: 28.2
Maximum Bitrate [Mbps]: 35.2
Use Maximum Render Quality
Use Previews
Import Into Project
Set Start Timecode
Render Alpha Channel Only
Time Interpolation: Frame Sampling
Estimated File Size: 383 MB
Metadata...
Queue
Export
Cancel
00:00:42:48
Fit
00:01:29:02
Source Range: Sequence In/Out

sample rate



channel on a video

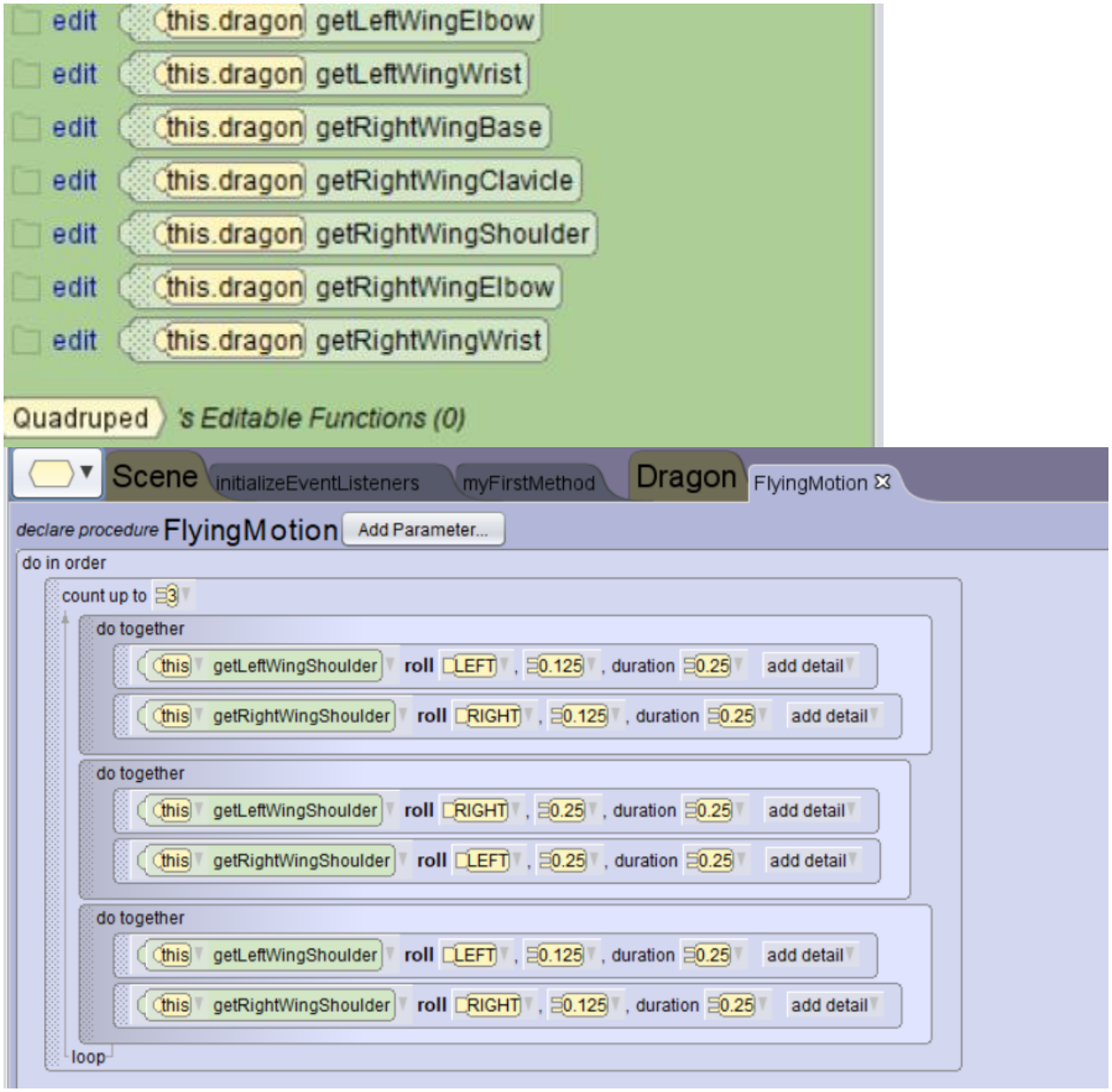
Lab 10:Animation Making

Link:

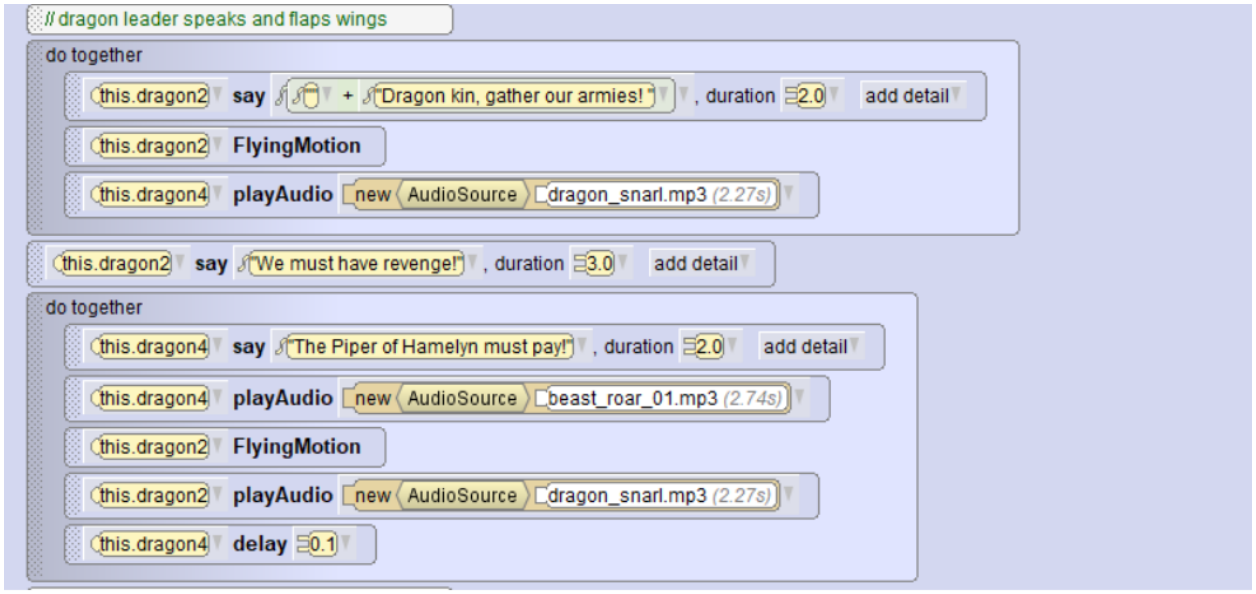
https://drive.google.com/file/d/1y8J90ff-6LjCdpmCw64xDgc-UuRfY94j/view?usp=sharing


Alice 3.4.0
File Edit Project Run Window Help
Starting Camera View
Run...
Edit Code
Undo
Redo
handle style:
Default
Rotation
Move
Resize
use snap
Snap details
this.firTreeTrunkSnowTallMirro
one shots
this.firTreeTrunkSnowTallMirror5's Proper
FirTreeTrunkSnowTallMirror
firTreeTrunkSnowTallMirr
Paint =
WHITE
Opacity =
1.0
Vehicle =
this
Position =
x: 27.50
y: -0.38
z: 27.70
Browse Gallery By Class Hierarchy
Browse Gallery By Theme
Browse Gallery By Group
Search Gallery
Shapes/Text
My Classes
All Classes
Quadruped classes
new Dragon(_)
new Dragon(DEFAULT_BLUE)
new Dragon(DEFAULT_GREEN)
new Dragon(DEFAULT_ORANGE)
new Dragon(DEFAULT_PURPLE)
new Dragon(DEFAULT_RED)
new Dragon(TUTU_BLUE)
Run...
Undo
Redo
handle style:
Default
Rotation
Move
Resize
use snap
Snap details
this.iceFloe
one shots
this.iceFloe's Properties
IceFloe
iceFloe
new
IceFloe
resource:
ICE_F
Paint =
WHITE
Opacity =
1.0


Scene
initializeEventListeners
myFirstMethod
this addSceneActivationListener
declare procedure sceneActivated
do in order
do together
this myFirstMethod
this playAudio new AudioSource epic.mp3 (66.67s)
Add Event Listener
Run...
Setup Scene
this.dragon
Procedures
Functions
group by category
Dragon 's Editable Functions (10)
edit this.dragon getLeftWingBase
edit this.dragon getLeftWingClavicle

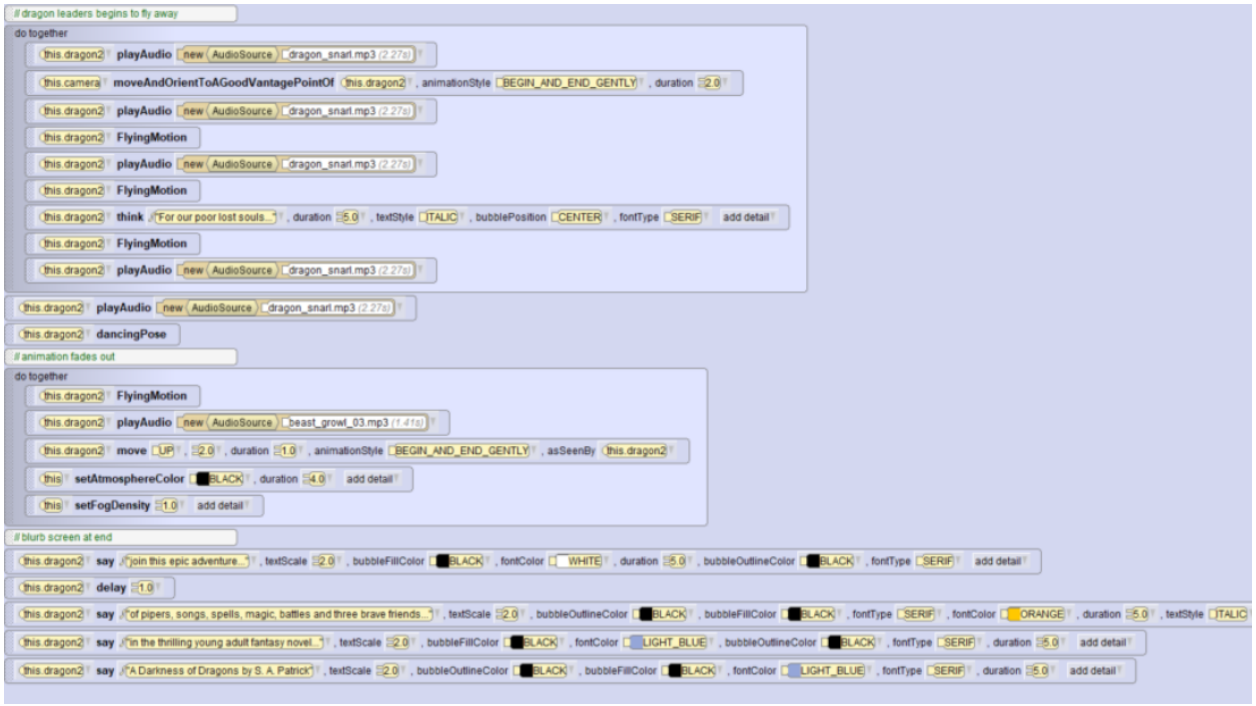

edit this.dragon getLeftWingElbow
edit this.dragon getLeftWingWrist
edit this.dragon getRightWingBase
edit this.dragon getRightWingClavicle
edit this.dragon getRightWingShoulder
edit this.dragon getRightWingElbow
edit this.dragon getRightWingWrist
Quadruped 's Editable Functions (0)
Scene initializeEventListeners myFirstMethod Dragon FlyingMotion
declare procedure FlyingMotion Add Parameter...
do in order
count up to 3
do together
this getLeftWingShoulder roll LEFT , 0.125 , duration 0.25 add detail
this getRightWingShoulder roll RIGHT , 0.125 , duration 0.25 add detail
do together
this getLeftWingShoulder roll RIGHT , 0.25 , duration 0.25 add detail
this getRightWingShoulder roll LEFT , 0.25 , duration 0.25 add detail
do together
this getLeftWingShoulder roll LEFT , 0.125 , duration 0.25 add detail
this getRightWingShoulder roll RIGHT , 0.125 , duration 0.25 add detail
loop


declare procedure myFirstMethod
do in order
// start the scene from an overhead camera marker
this.camera moveAndOrientTo this.cameraA , animationStyle BEGIN_AND_END_GENTLY , duration 0.25 add detail
this.dragon2 delay 1.0
// all of the dragons make a sound and one flaps wings and changes pose slightly
do together
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource beast_growl_03.mp3 (1.41s)
this.dragon4 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon4 delay 0.1
this.dragon3 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 dancingPose
// dragon leader speaks and flaps wings
do together
this.dragon2 say + "Dragon kin, gather our armies!" , duration 2.0 add detail
this.dragon2 FlyingMotion
this.dragon4 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 say "We must have revenge!" , duration 3.0 add detail


// dragon leader speaks and flaps wings
do together
this.dragon2 say + "Dragon kin, gather our armies!" , duration 2.0 add detail
this.dragon2 FlyingMotion
this.dragon4 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 say "We must have revenge!" , duration 3.0 add detail
do together
this.dragon4 say "The Piper of Hamelyn must pay!" , duration 2.0 add detail
this.dragon4 playAudio new AudioSource beast_roar_01.mp3 (2.74s)
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon4 delay 0.1


// two dragons fly away
do together
this.dragon4 FlyingMotion
this.dragon4 move BACKWARD , 10.0 , animationStyle BEGIN_AND_END_GENTLY , duration 2.0 , asSeenBy this.dragon2
this.dragon4 FlyingMotion
this.dragon2 FlyingMotion
this.dragon3 turn RIGHT , 0.125 , animationStyle BEGIN_AND_END_GENTLY , duration 0.5 add detail
this.dragon4 FlyingMotion
this.dragon3 move BACKWARD , 10.0 , animationStyle BEGIN_AND_END_GENTLY , asSeenBy this.dragon2 , duration 2.0
this.dragon3 FlyingMotion
// camera moves to zoom into dragon leader who is flapping wings
do together
this.dragon2 FlyingMotion
this.camera moveAndOrientTo this.cameraP_wingflap , animationStyle BEGIN_AND_END_GENTLY , pathStyle BEE_LINE , duration 3.0


// dragon leaders begins to fly away
do together
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.camera moveAndOrientToAGoodVantagePointOf this.dragon2 , animationStyle BEGIN_AND_END_GENTLY , duration 2.0
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 FlyingMotion
this.dragon2 think "For our poor lost souls..." , duration 5.0 , textStyle ITALIC , bubblePosition CENTER , fontType SERIF add detail
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 dancingPose
// animation fades out
do together


// dragon leaders begins to fly away
do together
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.camera moveAndOrientToAGoodVantagePointOf this.dragon2 , animationStyle BEGIN_AND_END_GENTLY , duration 2.0
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 FlyingMotion
this.dragon2 think "For our poor lost souls..." , duration 5.0 , textStyle ITALIC , bubblePosition CENTER , fontType SERIF add detail
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 playAudio new AudioSource dragon_snarl.mp3 (2.27s)
this.dragon2 dancingPose
// animation fades out
do together
this.dragon2 FlyingMotion
this.dragon2 playAudio new AudioSource beast_growl_03.mp3 (1.41s)
this.dragon2 move UP , 2.0 , duration 1.0 , animationStyle BEGIN_AND_END_GENTLY , asSeenBy this.dragon2
this setAtmosphereColor BLACK , duration 4.0 add detail
this setFogDensity 1.0 add detail
// blurb screen at end
this.dragon2 say "Join this epic adventure..." , textScale 2.0 , bubbleFillColor BLACK , fontColor WHITE , duration 5.0 , bubbleOutlineColor BLACK , fontType SERIF add detail
this.dragon2 delay 1.0
this.dragon2 say "of pipers, songs, spells, magic, battles and three brave friends..." , textScale 2.0 , bubbleOutlineColor BLACK , bubbleFillColor BLACK , fontType SERIF , fontColor ORANGE , duration 5.0 , textStyle ITALIC
this.dragon2 say "in the thrilling young adult fantasy novel..." , textScale 2.0 , bubbleFillColor BLACK , fontColor LIGHT_BLUE , bubbleOutlineColor BLACK , fontType SERIF , duration 5.0 add detail
this.dragon2 say "A Darkness of Dragons by S. A. Patrick" , textScale 2.0 , bubbleOutlineColor BLACK , bubbleFillColor BLACK , fontColor LIGHT_BLUE , fontType SERIF , duration 5.0 add detail