**LAB MANUAL OF**

**COMPUTERGRAPHICS**

**AND**

**MULTIMEDIA**

**ETCS 257**



Maharaja Agrasen Institute of Technology, PSP area, Sector – 22, Rohini, New Delhi – 110085

( Affiliated to Guru Gobind Singh Indraprastha University, New Delhi )

# 

Table of Contents

[1. Introduction to the Lab 1](#_Toc509404042)

[1.1 Lab Objective 1](#_Toc509404043)

[1.2 Course Outcomes 1](#_Toc509404044)

[2. LAB REQUIREMENTS 2](#_Toc509404045)

[3. LIST OF EXPERIMENTS (As prescribed by G.G.S.I.P.U) 3](#_Toc509404046)

[4. LIST OF EXPERIMENTS (Beyondthe syllabus) 4](#_Toc509404047)

[5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS 1](#_Toc509404048)

[6. MARKING SCHEME FOR THE PRACTICAL EXAMS 3](#_Toc509404049)

[7. INSTRUCTION FOR EACH LAB EXPERIMENT 5](#_Toc509404050)

[Experiment 1 5](#_Toc509404051)

[AIM 5](#_Toc509404052)

[DESCRIPTION 5](#_Toc509404053)

[VIVA QUESTIONS 9](#_Toc509404054)

[Experiment 2 10](#_Toc509404055)

[AIM 10](#_Toc509404056)

[ALGORITHMS 10](#_Toc509404057)

[DDA ALGORITHM FOR LINE DRAWING 10](#_Toc509404058)

[BRESENHAM’S LINE DRAWING ALGORITHM 10](#_Toc509404059)

[SAMPLE CODE 11](#_Toc509404060)

[DDA LINE DRAWING CODE 11](#_Toc509404061)

[BRESENHAM’S LINE DRAWING CODE 11](#_Toc509404062)

[SAMPLE INPUT-OUTPUT 12](#_Toc509404063)

[DDA ALGORITHM 12](#_Toc509404064)

[VIVA QUESTIONS 13](#_Toc509404065)

[EXPERIMENT 3 14](#_Toc509404066)

[AIM 14](#_Toc509404067)

[ALGORITHM 14](#_Toc509404068)

[BRESENHAM’S CIRCLE ALGORITHM 14](#_Toc509404069)

[MIDPOINT CIRCLE ALGORITHM 14](#_Toc509404070)

[SAMPLE CODE 17](#_Toc509404071)

[BRESENHAM’S CIRCLE ALGORITHM 17](#_Toc509404072)

[MIDPOINT CIRCLE ALGORITHM 17](#_Toc509404073)

[SAMPLE INPUT- OUTPUT 18](#_Toc509404074)

[VIVA QUESTIONS 19](#_Toc509404075)

[EXPERIMENT 4 20](#_Toc509404076)

[AIM 20](#_Toc509404077)

[ALGORITHMS 20](#_Toc509404078)

[TRANSLATION 20](#_Toc509404079)

[Rotation 21](#_Toc509404080)

[Scaling 22](#_Toc509404081)

[Reflection 23](#_Toc509404082)

[SAMPLE CODE 24](#_Toc509404083)

[PROGRAM TO SCALE THE TRIANGLE 24](#_Toc509404084)

[SAMPLE INPUT-OUTPUT 25](#_Toc509404085)

[VIVA QUESTIONS 26](#_Toc509404086)

[EXPERIMENT 5 27](#_Toc509404087)

[AIM 27](#_Toc509404088)

[ALGORITHM 27](#_Toc509404089)

[COHEN SUTHERLAND ALGORITHM 27](#_Toc509404090)

[SAMPLE CODE 28](#_Toc509404091)

[SAMPLE INPUT-OUTPUT 30](#_Toc509404092)

[VIVA QUESTIONS 31](#_Toc509404093)

[EXPERIMENT 6 32](#_Toc509404094)

[AIM 32](#_Toc509404095)

[ALGORITHM 32](#_Toc509404096)

[SAMPLE CODE 33](#_Toc509404097)

[SAMPLE INPUT-OUTPUT 37](#_Toc509404098)

[VIVA QUESTIONS 38](#_Toc509404099)

[EXPERIMENT 7 39](#_Toc509404100)

[AIM 39](#_Toc509404101)

[INSTRUCTIONS FOR MAYA 39](#_Toc509404102)

[VIVA QUESTIONS 60](#_Toc509404103)

[EXPERIMENT 8 61](#_Toc509404104)

[AIM 61](#_Toc509404105)

[INSTRUCTION 61](#_Toc509404106)

[VIVA QUESTIONS 63](#_Toc509404107)

[8. HINTS FOR PROGRAMS BEYOND SYLLABUS 64](#_Toc509404108)

[9. SAMPLE VIVA QUESTIONS 81](#_Toc509404109)

[10. SOME MORE QUESTIONS 85](#_Toc509404110)

# 

# 1. Introduction to the Lab

# 1.1 Lab Objective

This course aims to develop the understanding of concepts and algorithms of computer graphics using C Programming. Also this helps in understanding the working of MAYA tool for 2D/3D objects and their transformation using MAYA tool.

# 1.2 Course Outcomes

On successful completion of this Course, students should be able to:

1. Understand the use of C Graphics Library for writing the programs.
2. Implantation of scan conversion algorithms using C Programming.
3. Implementing the concept of 2D/3D transformation,
4. Implementing the concept of World & View Coordinate System and Clipping Algorithms.
5. Understand the object modeling and transformation using MAYA tool.

# 2. LAB REQUIREMENTS

|  |  |  |
| --- | --- | --- |
| **Hardware Detail** | Intel i3/C2D Processor/2 GB RAM/500GB HDD/MB/Lan Card/Key Board/ Mouse/CD Drive/15” Color Monitor/ UPS | 24 No. |
|  | LaserJet Printer | 1 No |
| **Software Detail** | Ubuntu Linux  MAYA |  |

# 3. LIST OF EXPERIMENTS(As prescribed by G.G.S.I.P.U)

**COMPUTER GRAPHICS & MULTIMEDIA LAB**

**Paper Code: ETCS-257**

**List of Experiments:**

1. Study of Fundamental Graphics Functions.

2. Implementation of Line drawing algorithms: DDA Algorithm, Bresenham's Algorithm

3. Implementation of Circle drawing algorithms: Bresenham's Algorithm, Mid Point Algorithm.

4. Programs on 2D and 3D transformations

5. Write a program to implement cohen Sutherland line clipping algorithm

6. Write a program to draw Bezier curve.

7. Using Flash/Maya perform different operations (rotation, scaling move etc..) on objects

8. Create a Bouncing Ball using Key frame animation and Path animation.

**NOTE:- At least 8 Experiments out of the list must be done in the semester.**

# 4. LIST OF EXPERIMENTS(Beyondthe syllabus)

**COMPUTER GRAPHICS & MULTIMEDIALAB**

|  |  |  |  |
| --- | --- | --- | --- |
| **CourseCode:**ETCS-257 |  |  |  |
| **Paper:**ComputerGraphics&MultimediaLab |  |  |  |

**List ofExperiments**

1. Write a programto rotatea circle(alternatively inside andoutside) around thecircumference ofanothercircle (animation).
2. Writeaprogramtodrawacarusinginbuildgraphicsfunctionandtranslateitfrombottom leftcorner to right bottom corner ofscreen (animation).
3. Writeaprogramtodrawballoonsusingin-builtgraphicsfunctionandtranslateitfrom bottom to top ofscreen (animation).
4. Writeaprogramtodrawacubeusinginbuildlibraryfunctionandperform3Dtransformations
   1. Translations in x,y,zdirections
   2. Rotation byangle450about zaxis,rotationby600abouty-axis in succession.
   3. Scalingin x-directionbyafactor of2, scaling iny- directionbyafactor of3.
5. Write aprogram to show animation ofa ballmovingin a helical path. (animation)
6. Write aprogram to show animation of solarsystem. (animation)
7. Create a Rainbow using graphics in C Program
8. Create a Digital clock using graphics in C Program. (animation)
9. Design a C Program for Tic Tac Toe Game. (game)

# 5. FORMAT OF THE LAB RECORD TO BE PREPARED BY THE STUDENTS

The front page of the lab record prepared by the students should have a cover page as displayed below.

***NAME OF THE LAB***

***Paper Code***

Font should be (Size 20”, italics bold, Times New Roman)

Faculty name Student name

Roll No.:

Semester:

Font should be (12”, Times Roman)



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

Font should be (18”, Times Roman)

**Index**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp. no** | **Experiment Name** | **Date of performance** | **Date of checking** | **Marks** | **Signature** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 6. MARKING SCHEME FOR THE PRACTICAL EXAMS

There will be two practical exams in each semester.

1. Internal Practical Exam
2. External Practical Exam

**INTERNAL PRACTICAL EXAM**

It is taken by the respective faculty of the batch.

**MARKING SCHEME FOR THIS EXAM IS**:

Total Marks: 40

Division of 10 marks per practical is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Rubrics for : Laboratory (General)** | | | |
| Sr No. | Experiment Component (LAC) | Max. Marks | Grading Rubrics | |
| 2 marks | 1 mark |
| 1 | Practical Performance | 2 | Completeness of practical, exhibits proficiency in using different types of inputs. | Incomplete practical, unformatted, lacks comments, Demonstrates no proficiency. |
| 2 | Output and Validation | 2 | Output is free of errors and output is obtained. Demonstrates excellent understanding of the concepts relevant to the experiment. | Output contains few logical errors and/or no output is obtained. Demonstrates partial understanding of the concepts relevant to the experiment. |
| 3 | Attendance and Viva Questions Answered | 4 | 1. Four marks for answering more than 75% questions.  2. Two marks for answering more then 50% questions.  3. One mark for answering less then 50% questions. | |
| 4 | Timely Submission of Lab Record | 2 | On time submission | Late submission |

Each experiment will be evaluated out of 10 marks. At the end of the semester average of 8 best performed practical will be considered as marks out of 40.

**EXTERNAL PRACTICAL EXAM**

It is taken by the concerned lecturer of the batch and by an external examiner. In this exam student needs to perform the experiment allotted at the time of the examination, a sheet will be given to the student in which some details asked by the examiner needs to be written and at the last viva will be taken by the external examiner.

**MARKING SCHEME FOR THIS EXAM IS**:

Total Marks: 60

Division of 60 marks is as follows

1. Sheet filled by the student: 20
2. Viva Voice: 15
3. Experiment performance: 15
4. File submitted: 10

**NOTE:**

* Internal marks + External marks = Total marks given to the students

(40 marks) (60 marks) (100 marks)

* Experiments given to perform can be from any section of the lab.

# 7. INSTRUCTION FOR EACH LAB EXPERIMENT

# Experiment 1

# AIM

Study of Fundamental Graphics Functions

# DESCRIPTION

The First lab of Computer Graphics aims at helping students understand the coding environment and the various commands that will be used in further labs. This study of fundamental commands help with understanding the various functions that are defined in graphics.h library.

1. **INITGRAPH**

Initializes the graphics system.

**Declaration**

void far initgraph(int far \*graphdriver)

**Remarks**

To start the graphic system, you must first call initgraph.

Initgraph initializes the graphic system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode.

Initgraph also resets all graphics settings (color, palette, current position, viewport, etc) to their defaults then resets graph.

1. **GETPIXEL, PUTPIXEL**

Get pixel gets the color of a specified pixel.

Put pixel places a pixel at a specified point.

**Decleration**

unsigned far getpixel(int x, int y)

void far putpixel (int x, int y, int color)

**Remarks**

Get pixel gets the color of the pixel located at(x,y);

Put pixel plots a point in the color defined at(x, y).

**Return value**

Get pixel returns the color of the given pixel.

Put pixel does not return any value.

1. **CLOSEGRAPH**

Shuts down the graphic system.

**Declaration**

void far closegraph(void);

**Remarks**

Closegraph de-allocates all memory allocated by the graphic system.

It then restores the screen to the mode it was in before you called initgraph.

**Return value**

None.

1. **ARC,CIRCLE,PIESLICE**

Arc draws a circular arc.

Circle draws a circle

Pieslice draws and fills a circular pie-slice

**Declaration**

Void far arc(int x,int y,int stangle,int end\_angle, int radius);

Void far circle(int x,int y, int radius);

Void far pieslice(int x,int y,int stangle,int end\_angle,int radius);

**Remarks**

Arc draws a circular arc in the current drawing color

Circle draws a circle in the current drawing color

Pieslice draws a pie-slice in the current drawing color, then fills it using the current fill pattern and fill color.

1. **ELLIPSE,FILLELIPSE,SECTOR**

Ellipse draws an elliptical arc.

Fillellipse draws and fills ellipse.

Sector draws and fills an elliptical pie-slice.

**Declaration**

Void far ellipse(int x,int y,int stangle,int end\_angle,int xradius,int yradius)

Void far fillellipse(int x,int y,int xradius,int yradius)

Void far sector(int x,int y,int stangle,int end\_angle,int xradius,int yradius)

**Remarks**

Ellipse draws an elliptical arc in the current drawing color.

Fillellipse draws an elliptical arc in the current drawing color and then fills it with fill color and fill pattern.

Sector draws an elliptical pie slice in the current drawing color and then fills it using the pattern and color defined by set fill style or set fill pattern.

1. **FLOODFILL**

Flood-fills a bounded region.

**Declaration**

Void far floodfill (int x,int y,int border)

**Remarks**

Flood fills an enclosed area on bitmap device.

The area bounded by the color border is flooded with the current fill pattern and fill color.

(x,y) is a―seed point. If the seed is within an enclosed area, the inside will be filled.

If the seed is outside the enclosed area, the exterior will be filled.

Use fillpoly instead of floodfill wherever possible so you can maintain code compatibility with future versions.

Flood fill does not work with the IBM-8514driver.

**Return value**

Ifanerroroccurswhilefloodingaregion,graphresultreturns‗1‘.

1. **GETCOLOR, SETCOLOR**

Get color returns the current drawing color.

Set color returns the current drawing color.

**Declaration**

int far getcolor(void);

void far setcolor(int color)

**Remarks**

Getcolor returns the current drawing color.

Setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.

To set a drawing color with setcolor, you can pass either the color number or the equivalent color name.

1. **LINE,LINEREL,LINETO**

Line draws a line between two specified points.

linerel draws a line relative distance from current position(CP).

Lineto draws a line from the current position(CP) to (x,y).

**Declaration**

void far line(int x1, int y1, int x2, int y2)

void far linerel (int dx, int dy)

void far lineto (int x, int y)

**Remarks**

Line draws a line from(x1, y1) to (x2, y2) using the current color, linestyle and thickness. It does not update the current position(CP).

Linerel draws a line from the CP to a point that is relative distance (dx,dy) from the CP, then advances the CP by(dx,dy).

Lineto draws a line from the CP to (x,y), then moves the CP to (x,y).

**Return value**

None

1. **RECTANGLE**

Draws a rectangle in graphics mode.

**Declaration**

Void far rectangle(int left,int top,int right,int bottom)

**Remarks**

It draws a rectangle in the current line style, thickness and drawing color.

(left,top) is the upper left corner of the rectangle, and (right,bottom) is its lower right corner.

# VIVA QUESTIONS

**Question 1. Define Computer Graphics.**

**Question 2. Can you tell which major components (hardware and software) are needed for computer graphics?**

**Question 3. What does refreshing of the screen mean?**

**Question 4. What is resolution?**

Question 5. What is Aspect Ratio?

# Experiment 2

# AIM

Implementation of Line drawing algorithms: DDA Algorithm, Bresenham's Algorithm

# ALGORITHMS

## DDA ALGORITHM FOR LINE DRAWING

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also declare gdriver=DETECT, gmode.
3. Input the two line end-points and store the left end-points in(x1,y1).
4. Load(x1, y1)into the frame buffer; that is , plot the first point. Put x=x1, y=y1.
5. Calculate dx=x2-x1 and dy=y2-y1.
6. If abs(dx)>abs(dy)
   1. s=abs(dx).
7. else
   1. s=abs(dy).
8. Thenxi=dx/sandyi=dy/s.
9. Startfromk=0andcontinuingtillk<s,thepointswillbe
   1. x=x+xi.
   2. y=y+yi.
   3. Placepixelsusingputpixelatpoints(x,y)inspecifiedcolour.
10. CloseGraphics mode.
11. Stop.

## BRESENHAM’S LINE DRAWING ALGORITHM

* 1. Start.
  2. Declare variable sx,y,x1,y1,x2,y2,p,dx,dy and also declare gdriver=DETECT,gmode.
  3. Initialize the graphic mode(gmode) with the path location
  4. Input the two line end-points and store the left end-points in(x1,y1).
  5. Load(x1,y1) into the frame-buffer; that is, plot the first point put x=x1,y=y1.
  6. Calculate dx=x2-x1 and dy=y2-y1, and obtain the initial value of decision parameter p as : p=(2dy-dx).
  7. Starting from first point (x,y) perform the following test:
  8. Repeat following steps while (x<=x2).
     1. If p<0 , next point is (x+1,y) and p=(p+2dy).
     2. Otherwise, the next point to plot is ( x+1,y+1 ) and p=(p+2dy-2dx).
     3. Place pixels using put pixelat points (x,y) in specified colour.
  9. Close Graphics mode.
  10. Stop

# SAMPLE CODE

## DDA LINE DRAWING CODE

voidDDA(intX0, intY0, intX1, intY1)

{

    // calculate dx & dy

    intdx = X1 - X0;

    intdy = Y1 - Y0;

    // calculate steps required for generating pixels

    intsteps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

    // calculate increment in x & y for each steps

    floatXinc = dx / (float) steps;

    floatYinc = dy / (float) steps;

    // Put pixel for each step

    floatX = X0;

    floatY = Y0;

    for(inti = 0; i <= steps; i++)

    {

        putpixel (X,Y,RED);  // put pixel at (X,Y)

        X += Xinc;           // increment in x at each step

        Y += Yinc;           // increment in y at each step

        delay(100);          // for visualization of line-

                             // generation step by step

    }

}

## BRESENHAM’S LINE DRAWING CODE

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

{

    int dx, dy, p, x, y;

    dx=x1-x0;

    dy=y1-y0;

    x=x0;

    y=y0;

    p=2\*dy-dx;

    while(x<x1)

    {

        if(p>=0)

        {

            putpixel(x,y,7);

            y=y+1;

            p=p+2\*dy-2\*dx;

        }

        else

        {

            putpixel(x,y,7);

            p=p+2\*dy;

        }

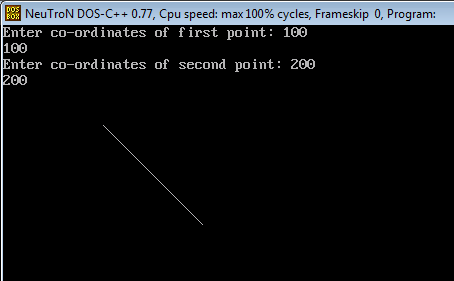
        x=x+1;

    }

}

# SAMPLE INPUT-OUTPUT

# DDA ALGORITHM



# VIVA QUESTIONS

Question 1. What is Line Drawing Algorithm ?

Question 2. Define getpixel and putpixel functions in graphics.h

Question 3. What is the difference between DDA algorithm and Bresenham’s Algorithm ?

Question 4. Give disadvantages of DDA algorithm.

Question 5. Give disadvantages of Bresenham’s algorithm.

# EXPERIMENT 3

# AIM

Implementation of Circle drawing algorithms: Bresenham's Algorithm, Mid Point Algorithm.

# ALGORITHM

## BRESENHAM’S CIRCLE ALGORITHM

**Bresenham Circle (Xc, Yc, R)**

**Description:** Here Xc and Yc denote the x – coordinate and y – coordinate of the center of the circle. R is the radius.

1. Set X = 0 and Y = R
2. Set D = 3-2R
3. Repeat While (X < Y)
   1. Call Draw Circle (Xc, Yc, X, Y)
   2. Set X = X+1
   3. If(D < 0) Then
   4. D = D+4X+6
   5. Else
   6. Set Y = Y – 1
   7. D = D+ 4(X-Y) + 10

[END IF]

1. Call Draw Circle (Xc, Yc, X, Y)
2. X++

[END WHILE]

1. EXIT

## MIDPOINT CIRCLE ALGORITHM

The equation of circle is X2+Y2=r2,X2+Y2=r2, where r is radius.

**Step 1** − Input radius **r** and circle center (xc,yc)(xc,yc) and obtain the first point on the circumference of the circle centered on the origin as

(x0, y0) = (0, r)

**Step 2** − Calculate the initial value of decision parameter as

P0P0 = 5/4 – r (See the following description for simplification of this equation.)

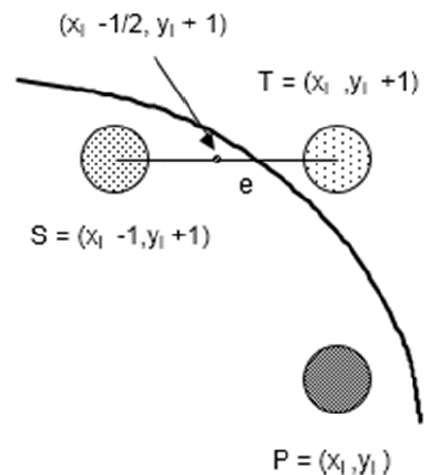
f(x, y)= x2+ y2- r2=0

f(xi-1/2+ e, yi+1)

=(xi-1/2+ e)2+(yi+1)2- r2

=(xi-1/2)2+(yi+1)2- r2+2(xi-1/2)e + e2

= f(xi-1/2, yi+1)+2(xi-1/2)e + e2=0



Let di= f(xi-1/2, yi+1)=-2(xi-1/2)e - e2

Thus,

If e <0then di >0 so choose point S =(xi-1, yi+1).

di+1= f(xi-1-1/2, yi+1+1)=((xi-1/2)-1)2+((yi+1)+1)2- r2

= di-2(xi-1)+2(yi+1)+1

= di+2(yi +1- xi +1)+1

if e >=0then di <=0 so choose point T =(xi, yi+1)

di+1= f(xi-1/2, yi+1+1)

= di+2yi+1+1

The initial value of di is

d0= f(r -1/2,0+1)=(r -1/2)2+12- r2

=5/4- r {1-r can be used if r is an integer}

When point S =(xi-1, yi+1)is chosen then

di+1= di+-2xi+1+2yi+1+1

When point T =(xi, yi+1)is chosen then

di+1= di+2yi+1+1

**Step 3** − At each XKXK position starting at K=0, perform the following test −

If PK<0thennext point on circle (0,0)is(XK+1,YK)and

PK+1= PK+2XK+1+1

Else

PK+1= PK+2XK+1+1–2YK+1

Where,2XK+1=2XK+2and2YK+1=2YK-2.

**Step 4** − Determine the symmetry points in other seven octants.

**Step 5** − Move each calculate pixel position (X, Y) onto the circular path centered on (XC,YC)(XC,YC) and plot the coordinate values.

X = X + XC, Y = Y + YC

**Step 6** − Repeat step-3 through 5 until X >= Y.

# SAMPLECODE

# BRESENHAM’S CIRCLE ALGORITHM

voiddrawCircle(intxc, intyc, intx, inty)

{

    putpixel(xc+x, yc+y, RED);

    putpixel(xc-x, yc+y, RED);

    putpixel(xc+x, yc-y, RED);

    putpixel(xc-x, yc-y, RED);

    putpixel(xc+y, yc+x, RED);

    putpixel(xc-y, yc+x, RED);

    putpixel(xc+y, yc-x, RED);

    putpixel(xc-y, yc-x, RED);

}

// Function for circle-generation

// using Bresenham's algorithm

voidcircleBres(intxc, intyc, intr)

{

    intx = 0, y = r;

    intd = 3 - 2 \* r;

    while(y >= x)

    {

        // for each pixel we will

        // draw all eight pixels

        drawCircle(xc, yc, x, y);

        x++;

        // check for decision parameter

        // and correspondingly

        // update d, x, y

        if(d > 0)

        {

            y--;

            d = d + 4 \* (x - y) + 10;

        }

        else

            d = d + 4 \* x + 6;

        drawCircle(xc, yc, x, y);

        delay(50);

    }

}

# MIDPOINT CIRCLE ALGORITHM

void drawcircle(int x0, int y0, int radius)

{

    int x = radius;

    int y = 0;

    int err = 0;

    while (x >= y)

    {

    putpixel(x0 + x, y0 + y, 7);

    putpixel(x0 + y, y0 + x, 7);

    putpixel(x0 - y, y0 + x, 7);

    putpixel(x0 - x, y0 + y, 7);

    putpixel(x0 - x, y0 - y, 7);

    putpixel(x0 - y, y0 - x, 7);

    putpixel(x0 + y, y0 - x, 7);

    putpixel(x0 + x, y0 - y, 7);

    if (err <= 0)

    {

        y += 1;

        err += 2\*y + 1;

    }

    if (err > 0)

    {

        x -= 1;

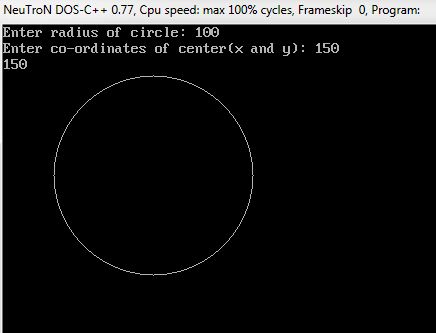
        err -= 2\*x + 1;

    }

    }

}

# SAMPLE INPUT-OUTPUT



# VIVA QUESTIONS

Question 1. What is 8 way symmetry ?

Question 2. Differentiate between midpoint circle algorithm and bresenham’s circle algorithm.

Question 3. Write the polar and Cartesian equation for Circle.

Question 4. How do we check whether a point lies on a circle or outside it ?

Question 5. Which is better algorithm for Circle drawing and why ?

# EXPERIMENT 4

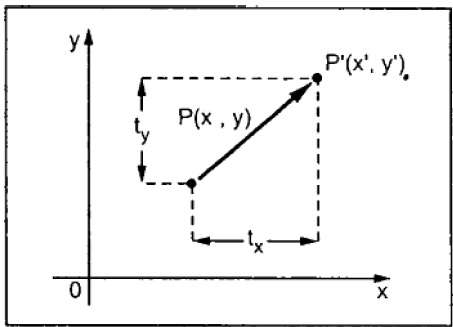
# AIM

Programs on 2D and 3D transformations

# ALGORITHMS

### TRANSLATION

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (tx, ty) to the original coordinate (X, Y) to get the new coordinate (X’, Y’).



From the above figure, you can write that −

**X’ = X + tx**

**Y’ = Y + ty**

The pair (tx, ty) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

P=[X][Y]P=[X][Y] p' = [X′][Y′][X′][Y′]T = [tx][ty][tx][ty]

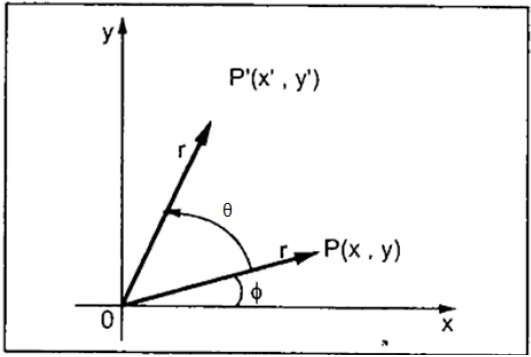
We can write it as −

**P’ = P + T**

### Rotation

In rotation, we rotate the object at particular angle θ (theta) from its origin. From the following figure, we can see that the point P(X, Y) is located at angle φ from the horizontal X coordinate with distance r from the origin.

Let us suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P’ (X’, Y’).



Using standard trigonometric the original coordinate of point P(X, Y) can be represented as −

X=rcosϕ......(1)X=rcosϕ......(1)

Y=rsinϕ......(2)Y=rsinϕ......(2)

Same way we can represent the point P’ (X’, Y’) as −

x′=rcos(ϕ+θ)=rcosϕcosθ−rsinϕsinθ.......(3)x′=rcos(ϕ+θ)=rcosϕcosθ−rsinϕsinθ.......(3)

y′=rsin(ϕ+θ)=rcosϕsinθ+rsinϕcosθ.......(4)y′=rsin(ϕ+θ)=rcosϕsinθ+rsinϕcosθ.......(4)

Substituting equation (1) & (2) in (3) & (4) respectively, we will get

x′=xcosθ−ysinθx′=xcosθ−ysinθ

y′=xsinθ+ycosθy′=xsinθ+ycosθ

Representing the above equation in matrix form,

[X′Y′]=[XY][cosθ−sinθsinθcosθ]OR[X′Y′]=[XY][cosθsinθ−sinθcosθ]OR

P’ = P . R

Where R is the rotation matrix

R=[cosθ−sinθsinθcosθ]R=[cosθsinθ−sinθcosθ]

The rotation angle can be positive and negative.

For positive rotation angle, we can use the above rotation matrix. However, for negative angle rotation, the matrix will change as shown below −

R=[cos(−θ)−sin(−θ)sin(−θ)cos(−θ)]R=[cos(−θ)sin(−θ)−sin(−θ)cos(−θ)]

=[cosθsinθ−sinθcosθ](∵cos(−θ)=cosθandsin(−θ)=−sinθ)=[cosθ−sinθsinθcosθ](∵cos(−θ)=cosθandsin(−θ)=−sinθ)

### Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

Let us assume that the original coordinates are (X, Y), the scaling factors are (SX, SY), and the produced coordinates are (X’, Y’). This can be mathematically represented as shown below −

**X' = X . SX and Y' = Y . SY**

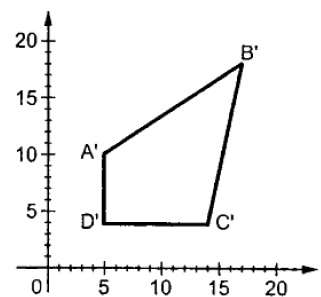
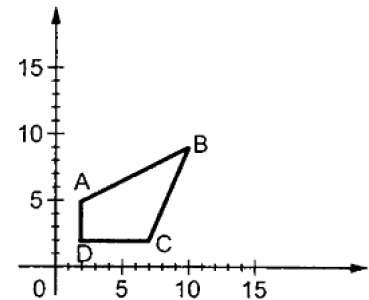
The scaling factor SX, SY scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below −

(X′Y′)=(XY)[Sx00Sy](X′Y′)=(XY)[Sx00Sy]

OR

**P’ = P . S**

Where S is the scaling matrix. The scaling process is shown in the following figure.

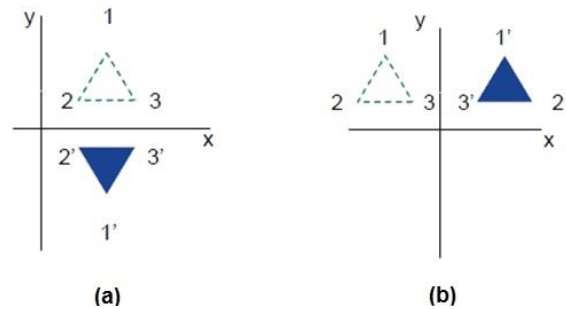


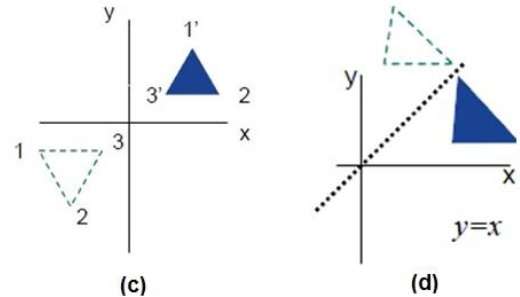
If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

### Reflection

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180°. In reflection transformation, the size of the object does not change.

The following figures show reflections with respect to X and Y axes, and about the origin respectively.





# 

# SAMPLECODE

### PROGRAM TO SCALE THE TRIANGLE

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

intmain()

{

int gd=DETECT,gm;

int x1,y1,x2,y2,x3,y3,x4,y4;floatsx,sy;

printf("Enter thefirst coordinates of triangle\n");scanf(%d%d,&x1,&y1);

printf("Enter thesecondcoordinates of triangle\n");scanf(%d%d,&x2,&y2);

printf"Enter thethird coordinates of triangle\n");scanf(%d%d,&x3,&y3);

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

int poly[8]={x1,y1,x2,y2,x3,y3,x1,y1};

drawpoly(4,poly);

getch();

printf("Enter thescalingfactors\n");scanf(%d%d,&sx,&sy);

x4=sx\*x1-x1;

y4=sy\*y1-y1;

x1=sx\*x1-x4;y1=sy\*y1-y4;x2=sx\*x2-x4;y2=sy\*y2-y4;x3=sx\*x3-x4;y3=sy\*y3-y4;poly[0]=x1;poly[1]=y1;poly[2]=x2;poly[3]=y2;poly[4]=x3;poly[5]=y3;poly[6]=x1;poly[7]=y1;

cleardevice();

drawpoly(4,poly);

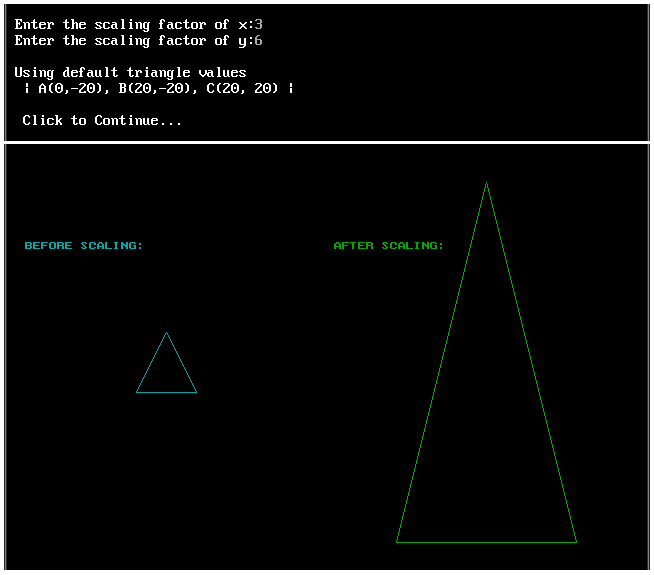
getch();

closegraph();

return 0;

}

# SAMPLE INPUT-OUTPUT



# VIVA QUESTIONS

Question 1. Translate the square ABCD whose co-ordinate are A(0,0), b(3,0), C(3,3), D(0,3) by 2 units in both direction and then scale it by 1.5 units in x direction and 0.5 units in y direction.

Question 2. Perform a 45o rotation of a triangle A(0,0, B(1,1), C(5,2)

1) About the origin.

2) About the point p(-1,-1)

Question 3. Find the transformation matrix that transforms the square ABCD whose center is at (2,2) is reduced to half of its size, with center still remaining at (2,2). The coordinate of square ABCD are A(0,0), B(0,4), C(4,4) and D(4,0). Find the co-ordinate of new square.

Question 4. Consider the square A(1,0), B(0,0), C(0,1), D(1,1). Rotate the square ABCD by 45o

clockwise about A(1,0).

Question 5. Magnify the triangle with vertices A(0,0), B(1,1) and C(5,2) to twice its size while keeping C(5,2) fixed.

# EXPERIMENT 5

# AIM

Write a program to implement Cohen-Sutherland line clipping algorithm

# ALGORITHM

## COHEN SUTHERLAND ALGORITHM

Step 1 : Assign a region code for two endpoints of given line.

Step 2 : If both endpoints have a region code 0000

then given line is completely inside.

Step 3 : Else, perform the logical AND operation for both region codes.

Step 3.1 : If the result is not 0000, then given line is completely

outside.

Step 3.2 : Else line is partially inside.

Step 3.2.1 : Choose an endpoint of the line

that is outside the given rectangle.

Step 3.2.2 : Find the intersection point of the

rectangular boundary (based on region code).

Step 3.2.3 : Replace endpoint with the intersection point

and update the region code.

Step 3.2.4 : Repeat step 2 until we find a clipped line either

trivially accepted or trivially rejected.

Step 4 : Repeat step 1 for other lines

# SAMPLECODE

**enum** { TOP = 0x1, BOTTOM = 0x2, RIGHT = 0x4, LEFT = 0x8 };

outcode compute\_outcode(int x, int y,

int xmin, int ymin, int xmax, int ymax)

{

outcode oc = 0;

if (y > ymax)

oc |= TOP;

else if (y < ymin)

oc |= BOTTOM;

if (x > xmax)

oc |= RIGHT;

else if (x < xmin)

oc |= LEFT;

return oc;

}

void cohen\_sutherland (double x1, double y1, double x2, double y2,

double xmin, double ymin, double xmax, double ymax)

{

int accept;

int done;

outcode outcode1, outcode2;

  accept = FALSE;

done = FALSE;

  outcode1 = compute\_outcode (x1, y1, xmin, ymin, xmax, ymax);

outcode2 = compute\_outcode (x2, y2, xmin, ymin, xmax, ymax);

do

{

if (outcode1 == 0 && outcode2 == 0)

{

accept = TRUE;

done = TRUE;

}

else if (outcode1 & outcode2)

{

done = TRUE;

}

else

{

double x, y;

int outcode\_ex = outcode1 ? outcode1 : outcode2;

if (outcode\_ex & TOP)

{

x = x1 + (x2 - x1) \* (ymax - y1) / (y2 - y1);

y = ymax;

}

else if (outcode\_ex & BOTTOM)

{

x = x1 + (x2 - x1) \* (ymin - y1) / (y2 - y1);

y = ymin;

}

else if (outcode\_ex & RIGHT)

{

y = y1 + (y2 - y1) \* (xmax - x1) / (x2 - x1);

x = xmax;

}

else

{

y = y1 + (y2 - y1) \* (xmin - x1) / (x2 - x1);

x = xmin;

}

if (outcode\_ex == outcode1)

{

x1 = x;

y1 = y;

outcode1 = compute\_outcode (x1, y1, xmin, ymin, xmax, ymax);

}

else

{

x2 = x;

y2 = y;

outcode2 = compute\_outcode (x2, y2, xmin, ymin, xmax, ymax);

}

}

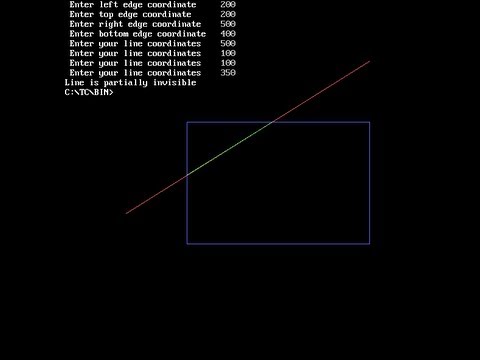
} while (done == FALSE);

  if (accept == TRUE)

line (x1, y1, x2, y2);

}

## SAMPLE INPUT-OUTPUT



# VIVA QUESTIONS

QUESTION 1. What is Rasterization ?

QUESTION 2. A rectangular clipping window is defined by the following window co- ordinates: (0; 0) for the left, bottom corner and (5; 4) for the right, top corner. We are also given two line segments: Line AB (from A(-1;-1) to B(6; 6)) and Line CD (from C(-1; 1) to D(4;-3)) that we want to clip against the window using the Cohen-Sutherland Clipping algorithm. What is the sequence of bit-codesgenerated by the algorithm when it is run on the lines AB and CD. Also, mention what is the final result of the clipping.

The 4 bit bitcode PQRS are defined as per the following convention:

P is 1 if x <0

Q is 1 if x >5

R is 1 if y <0

S is 1 if y >4

QUESTION 3. Differentiate between Vector and Scalar Graphics.

QUESTION 4. What is the need for Clipping Algorithms?

QUESTION 5. List disadvantages of Vector graphics.

# EXPERIMENT 6

# AIM

WRITE A PROGRAM TO DRAW A BEZIER CURVE

ALGORITHM

1. Start.  
2. Read the number of control points(no) and set the number n=no-1  
3. for i as 0 to n do step 5 and 6  
4. Read the end point to xk[i] and yk[i]  
5. Set i=i+1  
6. Set cpx=xk[0] and cpy=yk[0]  
7. for u is 0.1 to 0.3 repeat as follows  
8. Setxu as 0 and yu as 0  
9. Fori =n to 0 repeat as follows  
10. If i is not equal 0 then set xu=xu+xk[i]\*Bezier(n,i) \* u^i\*(1-u)^n-i  
 Else

set xu=xu+xk[i]\*Bezier(n,i)\*(1-u)^n-i  
11. Fori=n to 0 do  
12. If i is not equal to 0 then set yu=yu+yk[i]\*Bezier(n,i)\*u^i(1-u)^i-1  
 Else

set yu=yu+yk[i]\*Bezier(n,i) (1-u)^n-i  
13. Draw the curve based on the values calculated  
14. Input the control points and interval d. set n=d-1  
15. Inside the Bezier function calculate n!/(i!)\*(n-1)! By calling the factorial function for each variable.  
16. In the factorial function, initialize fact=1 and get the number into i.  
17. Calculate fact=fact\*i recursively and decrement i till 1 is reached.  
18. End.

# SAMPLECODE

/ C program to implement

// Cubic Bezier Curve

/\* install SDL library for running thing code\*/

/\* install by using this commamnd line : sudo apt-get install libsdl2-dev \*/

/\* run this code using command : gcc fileName.c -lSDL2 -lm\*/

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<SDL2/SDL.h>

SDL\_Window\* window = NULL;

SDL\_Renderer\* renderer = NULL;

int mousePosX , mousePosY ;

int xnew , ynew ;

/\*Function to draw all other 7 pixels present at symmetric position\*/

void drawCircle(int xc, int yc, int x, int y)

{

SDL\_RenderDrawPoint(renderer,xc+x,yc+y) ;

SDL\_RenderDrawPoint(renderer,xc-x,yc+y);

SDL\_RenderDrawPoint(renderer,xc+x,yc-y);

SDL\_RenderDrawPoint(renderer,xc-x,yc-y);

SDL\_RenderDrawPoint(renderer,xc+y,yc+x);

SDL\_RenderDrawPoint(renderer,xc-y,yc+x);

SDL\_RenderDrawPoint(renderer,xc+y,yc-x);

SDL\_RenderDrawPoint(renderer,xc-y,yc-x);

}

/\*Function for circle-generation using Bresenham's algorithm \*/

void circleBres(int xc, int yc, int r)

{

int x = 0, y = r;

int d = 3 - 2 \* r;

while (y >= x)

{

/\*for each pixel we will draw all eight pixels \*/

drawCircle(xc, yc, x, y);

x++;

/\*check for decision parameter and correspondingly update d, x, y\*/

if (d > 0)

{

y--;

d = d + 4 \* (x - y) + 10;

}

else

d = d + 4 \* x + 6;

drawCircle(xc, yc, x, y);

}

}

/\* Function that take input as Control Point x\_coordinates and

Control Point y\_coordinates and draw bezier curve \*/

void bezierCurve(int x[] , int y[])

{

double xu = 0.0 , yu = 0.0 , u = 0.0 ;

int i = 0 ;

for(u = 0.0 ; u <= 1.0 ; u += 0.0001)

{

xu = pow(1-u,3)\*x[0]+3\*u\*pow(1-u,2)\*x[1]+3\*pow(u,2)\*(1-u)\*x[2]

+pow(u,3)\*x[3];

yu = pow(1-u,3)\*y[0]+3\*u\*pow(1-u,2)\*y[1]+3\*pow(u,2)\*(1-u)\*y[2]

+pow(u,3)\*y[3];

SDL\_RenderDrawPoint(renderer , (int)xu , (int)yu) ;

}

}

int main(int argc, char\* argv[])

{

/\*initialize sdl\*/

if (SDL\_Init(SDL\_INIT\_EVERYTHING) == 0)

{

/\*

This function is used to create a window and default renderer.

int SDL\_CreateWindowAndRenderer(int width

,int height

,Uint32 window\_flags

,SDL\_Window\*\* window

,SDL\_Renderer\*\* renderer)

return 0 on success and -1 on error

\*/

if(SDL\_CreateWindowAndRenderer(640, 480, 0, &window, &renderer) == 0)

{

SDL\_bool done = SDL\_FALSE;

int i = 0 ;

int x[4] , y[4] , flagDrawn = 0 ;

while (!done)

{

SDL\_Event event;

/\*set background color to black\*/

SDL\_SetRenderDrawColor(renderer, 0, 0, 0, SDL\_ALPHA\_OPAQUE);

SDL\_RenderClear(renderer);

/\*set draw color to white\*/

SDL\_SetRenderDrawColor(renderer, 255, 255, 255, SDL\_ALPHA\_OPAQUE);

/\* We are drawing cubic bezier curve

which has four control points \*/

if(i==4)

{

bezierCurve(x , y) ;

flagDrawn = 1 ;

}

/\*grey color circle to encircle control Point P0\*/

SDL\_SetRenderDrawColor(renderer, 128, 128, 128, SDL\_ALPHA\_OPAQUE);

circleBres(x[0] , y[0] , 8) ;

/\*Red Line between control Point P0 & P1\*/

SDL\_SetRenderDrawColor(renderer, 255, 0, 0, SDL\_ALPHA\_OPAQUE);

SDL\_RenderDrawLine(renderer , x[0] , y[0] , x[1] , y[1]) ;

/\*grey color circle to encircle control Point P1\*/

SDL\_SetRenderDrawColor(renderer, 128, 128, 128, SDL\_ALPHA\_OPAQUE);

circleBres(x[1] , y[1] , 8) ;

/\*Red Line between control Point P1 & P2\*/

SDL\_SetRenderDrawColor(renderer, 255, 0, 0, SDL\_ALPHA\_OPAQUE);

SDL\_RenderDrawLine(renderer , x[1] , y[1] , x[2] , y[2]) ;

/\*grey color circle to encircle control Point P2\*/

SDL\_SetRenderDrawColor(renderer, 128, 128, 128, SDL\_ALPHA\_OPAQUE);

circleBres(x[2] , y[2] , 8) ;

/\*Red Line between control Point P2 & P3\*/

SDL\_SetRenderDrawColor(renderer, 255, 0, 0, SDL\_ALPHA\_OPAQUE);

SDL\_RenderDrawLine(renderer , x[2] , y[2] , x[3] , y[3]) ;

/\*grey color circle to encircle control Point P3\*/

SDL\_SetRenderDrawColor(renderer, 128, 128, 128, SDL\_ALPHA\_OPAQUE);

circleBres(x[3] , y[3] , 8) ;

/\*We are Polling SDL events\*/

if (SDL\_PollEvent(&event))

{

/\* if window cross button clicked then quit from window \*/

if (event.type == SDL\_QUIT)

{

done = SDL\_TRUE;

}

/\*Mouse Button is Down \*/

if(event.type == SDL\_MOUSEBUTTONDOWN)

{

/\*If left mouse button down then store

that point as control point\*/

if(event.button.button == SDL\_BUTTON\_LEFT)

{

/\*store only four points

because of cubic bezier curve\*/

if(i < 4)

{

printf("Control Point(P%d):(%d,%d)\n"

,i,mousePosX,mousePosY) ;

/\*Storing Mouse x and y positions

in our x and y coordinate array \*/

x[i] = mousePosX ;

y[i] = mousePosY ;

i++ ;

}

}

}

/\*Mouse is in motion\*/

if(event.type == SDL\_MOUSEMOTION)

{

/\*get x and y postions from motion of mouse\*/

xnew = event.motion.x ;

ynew = event.motion.y ;

int j ;

/\* change coordinates of control point

after bezier curve has been drawn \*/

if(flagDrawn == 1)

{

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

{

/\*Check mouse position if in b/w circle then

change position of that control point to mouse new

position which are coming from mouse motion\*/

if((float)sqrt(abs(xnew-x[j]) \* abs(xnew-x[j])

+ abs(ynew-y[j]) \* abs(ynew-y[j])) < 8.0)

{

/\*change coordinate of jth control point\*/

x[j] = xnew ;

y[j] = ynew ;

printf("Changed Control Point(P%d):(%d,%d)\n"

,j,xnew,ynew) ;

}

}

}

/\*updating mouse positions to positions

coming from motion\*/

mousePosX = xnew ;

mousePosY = ynew ;

}

}

/\*show the window\*/

SDL\_RenderPresent(renderer);

}

}

/\*Destroy the renderer and window\*/

if (renderer)

{

SDL\_DestroyRenderer(renderer);

}

if (window)

{

SDL\_DestroyWindow(window);

}

}

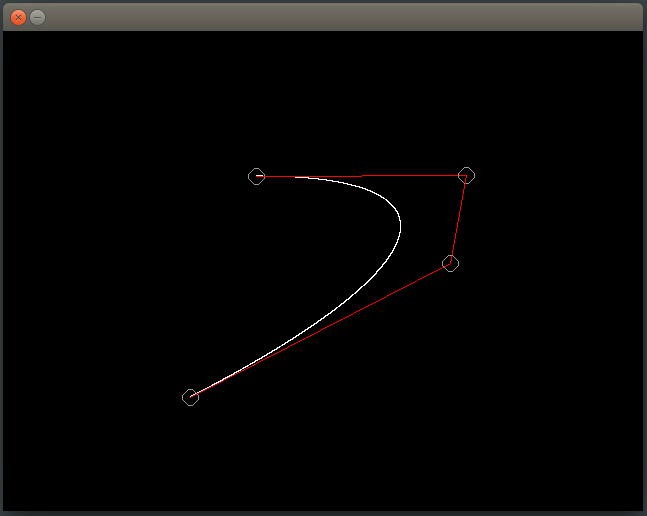
/\*clean up SDL\*/

SDL\_Quit();

return 0;

}

# SAMPLE INPUT-OUTPUT



# VIVA QUESTIONS

QUESTION 1. What do you mean by Curve ?

QUESTION 2. How many types of Curves have you studied ?

QUESTION 3. What are B-Splines ?

QUESTION 4. Difference between B-Splines and Bezier Curves ?

QUESTION 5. What are control points in a curve ?

# EXPERIMENT 7

# AIM

Using Flash/Maya perform different operations (rotation, scaling move etc..) on objects

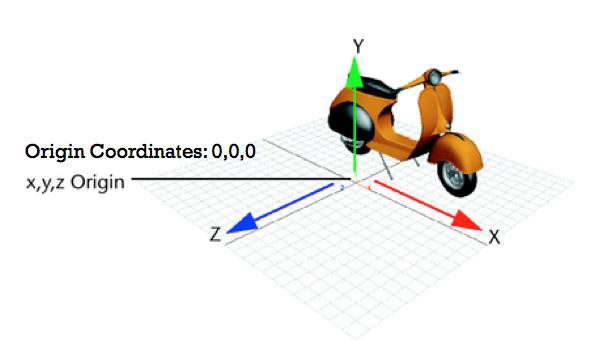
# INSTRUCTIONS FOR MAYA

Maya, is a 3D computer graphics application that runs on Windows, macOS and Linux, originally developed by Alias Systems Corporation (formerly Alias|Wavefront) and currently owned and developed by Autodesk, Inc. It is used to create interactive 3D applications, including video games, animated film, TV series, or visual effects.

Movingfrom2Dto3Ddesigncanbechallengingsincemostofusdesignusingonlytwodimensions.However,you‘vedonethisbeforebutyoujustdon‘tremember!Maybeyouhaven‘trealizedthatbuildingobjectsinMaya3DissomehowsimilartobuildingobjectsusingLegos.ThekeytodesigninMaya3Distothinkspatially.InMaya,wedesignnotonlyacknowledgingwidth and heightwe alsoincorporate depth!

Ittakessomepracticeandlotsofpatiencetodevelop3Dspatialability,sobelowyou‘llfindsome basicMaya 101concepts that will helpyoubetterunderstand the3Denvironment.

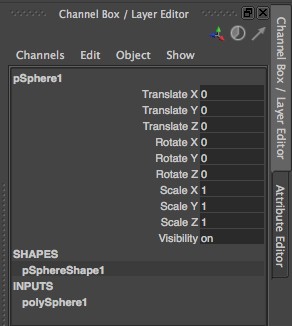
#### X,Yand Z



InMayatheXaxisisthewidth(red),theyaxisistheheight(green)andtheZaxisisthedepth(blue).They-axisisalsoreferredtoasY-up.Thecenterofthecoordinatesystemiscalledtheoriginand the coordinates are 0, 0, 0 for x,y,z.

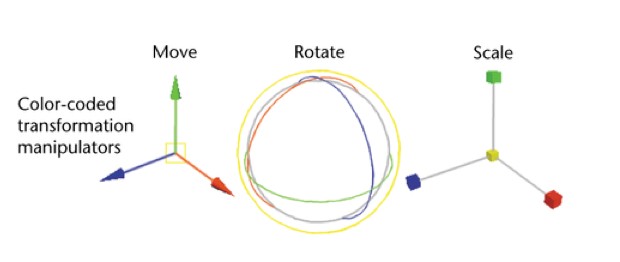
Ifyouwant to manipulate anobject in Mayayoucan:

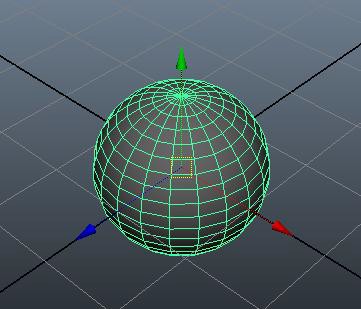
1. Enterx,yandzvaluesfortranslate,rotateandscalein―ChannelBox



* + Translate:Value―0inx,yandz,positiontheobjectonthecenteror―origin‖oftheworkspace.
  + Rotate: Ifyoubymistakerotateanobject,setallthevaluesto―0.Thiswilltaketheobjectback to its default position.
  + Scale:―1isthedefaultvalueanditmeanstheobject‘ssizeis100%.Ifyouwanttomaketheobjecttwiceitssizeenter2inallthescalevalues.Ifyouwanttomakeanobjecthalfits size enter0.5 in all the scale values

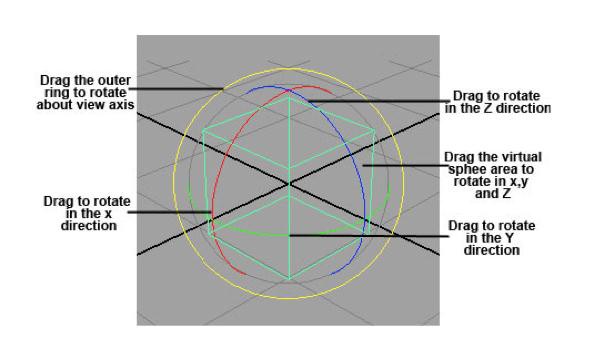
1. Usethe manipulatorshandles ofthe movetool,scale tool and rotatetool.



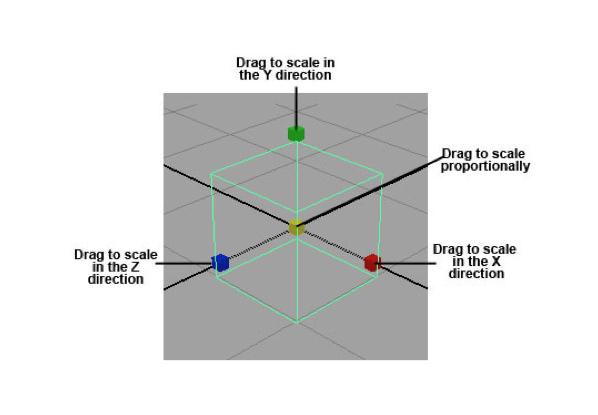


Whenmoving&rotatingobjectsinMaya,it‘srecommendedthatyoumoveorrotatetheobjectonlyalongoneaxis, thiscanhelpyou maintainyour objectsaligned.

You cando this bypullingthe arrowsof themovetool or bydragging therings oftherotatetool.Movingtheobjectsfromthecenterhandlewillallowyoutomovetheobjectfreelyacrosstheviewplaneand it can behard to control its translationandrotation.



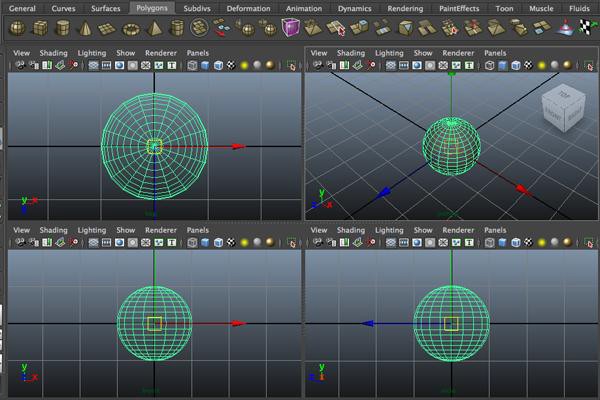
WhenscalingobjectisMaya,ifyouwanttomaintaintheobject‘sproportionsdragthecenterboxto scale uniformlyin alldirections.



#### Views

Whenworkingina3Denvironmentisveryimportanttocheckthattheobjectyouarebuildinglooksgoodinalltheviews.Lookingtojustoneviewcanbedeceivingandyouobjectmightnotbe positionedcorrectly.

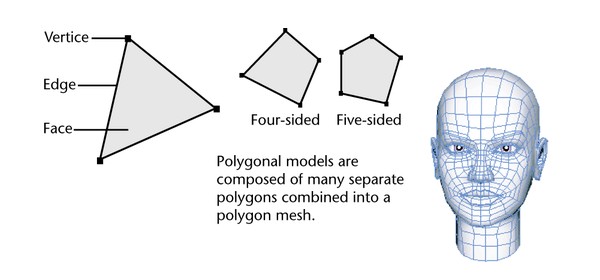
InordertoeasethedesignprocessMayaprovidesupto4views,oneperspectiveviewandthreeorthographicviews.



Whenyoulookattheobjectfromtheperspectiveviewyoucanrevolvethecameratofreelytumblearoundtheobject.However,whenyouuseorthographicviews(top,side&front)theyonlyfocus in 2axes at thetime, soyoucan‘t tumble around (unlessyou usethe tumbletool).

#### Polygonal Modeling vs NURBSModelingPolygons:

PolygonsarethemostbasicgeometryinMayatheyarestraight-sidedshapesof3sidesormore.Polygonshavefaces,edges&vertices,someexamplesofpolygonsorprimitiveshapesarespheres, cubes, cylinders,cones, andplanes.

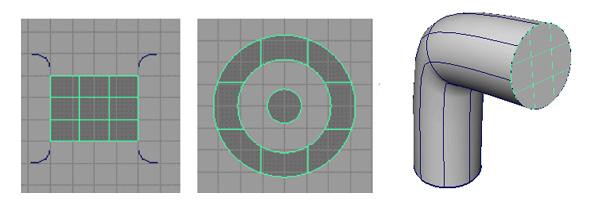


Becausepolygon‘ssurfacescanbedirectlyextruded,scaledandpositioneddesignerspreferthem to createcharactersin Maya.

#### NURBSCurves and NURBSSurfaces:

NURBScurvesstandforNon-UniformRationalB-Splines.NURBScurvesareveryhandysinceyou can draw them in Mayaor import them froma vector program like AdobeIllustrator.

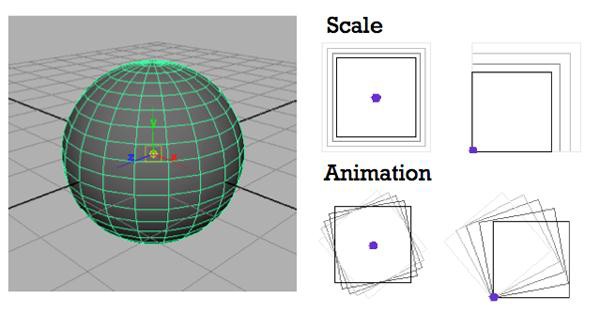
NURBSprimitivesorsurfaces,useUVcoordinatespaceandyoucanmodifythembytrimmingawayportionsoftheirforms,bevelingtheiredges,orbysculptingthemintodifferentshapesusingtheMaya Artisan sculptingtools



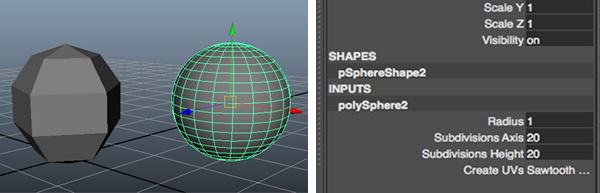
NURBSarepreferredforconstructingorganic3Dformsbecauseoftheirsmooth&naturalcharacteristics.

**Pivot**

Alltransformationstoanobjectarerelativetothepivotpoint.Theobject‘spivotaffectsscalesincetheobjectwillscaleoutfromorintowardthepivotpoint.Thepivotalsoaffectstheobject‘srotation/animationbecauseanobjectrotates around the pivot point.



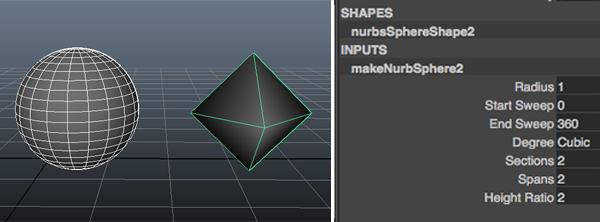
#### SubdivisionsPolygonSubdivisions:



Ifapolygonhaslesssubdivisions,forexampleanvalueof―5‖,itwillhavelessdetailandaroughersurface

Ifapolygonhasmoresubdivisions,forexampleanvalueof―20‖,itwillhavemoredetail&asmoothersurface

#### NURBSSubdivisions

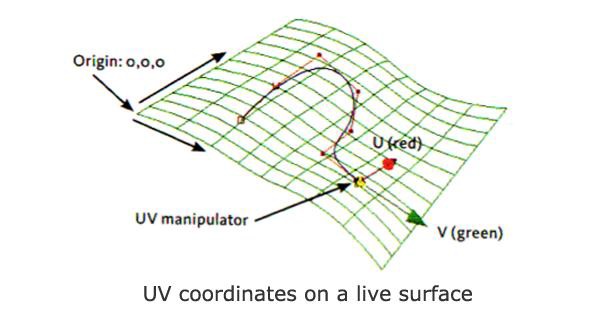


IfaNURBhaslesssections&spans,forexampleanvalueof―2,itwillhavelessdetailandaroughersurface

IfaNURBhasmoresections&spans,forexampleanvalueof―20,itwillhavemoredetail&asmoother surface

#### UV Coordinate Space:

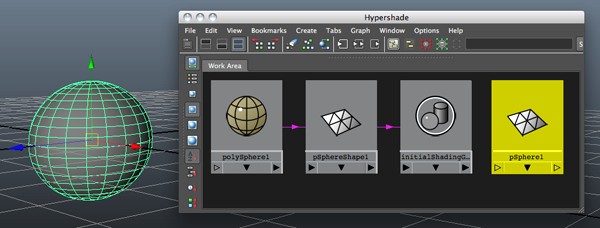
Surfacesin Mayahave their own coordinatespace.UV Coordinate Spaceis useful whenworking with curve-on-surfaces andpositioningtextures.



#### Nodes

EveryelementinMayaisbuiltwithasingleoraseriesofnodes.Nodesdefineallattributeslikelighting, shading andgeometry.

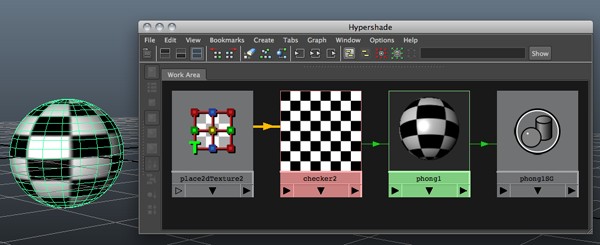
Forexampleaprimitivepolygon,suchasasphere,isbuiltfromseveralnodes:acreationnodethatrecordstheoptionsthatcreatedthesphere,atransformnodethatrecordshowtheobjectismoved,rotated,andscaled,andashapenodethatstoresthepositionsofthespherescontrolpoints.



#### Shape nodes:

Holdsanobject‘sgeometryattributesorattributesotherthantheobject‘stransformnodeattributes.

#### Renderingnodes:

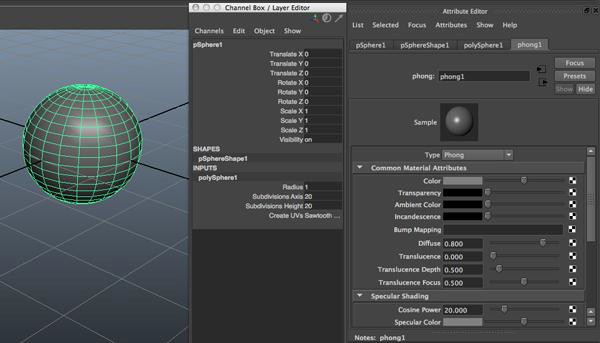


Materialsandtextureseachhavenodescontainingattributesthatcontroltheirlook.Textureplacementnodes haveattributes that control how atextureis fitted onto a surface.

#### Attributes:

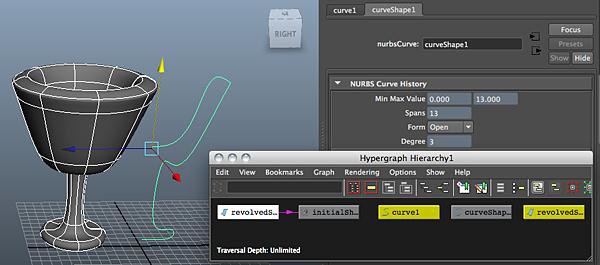
Anattributeisapositionassociatedwithanodethatcanholdavalueoraconnectiontoanothernode.Attributescontrolhow anodeworks.

To changeanobject‘sattributesyoucan usetheAttributeEditor &the Channel Box.

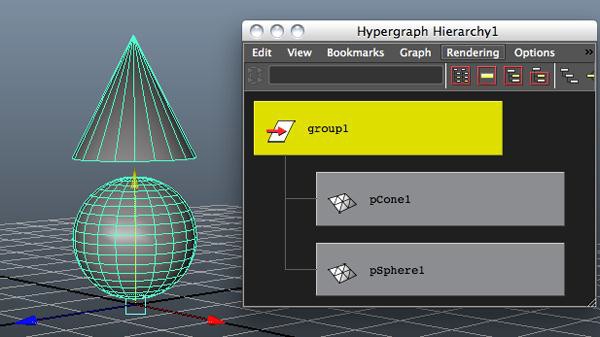


#### ConstructionHistory:

WhenyouworkinMaya,alltheactionsyoumakecreatenodesintheconstructionhistoryoftheobjectsyouareworkingon.Constructionhistoryisveryimportantbecauseitallowsyoutochangeanobject.

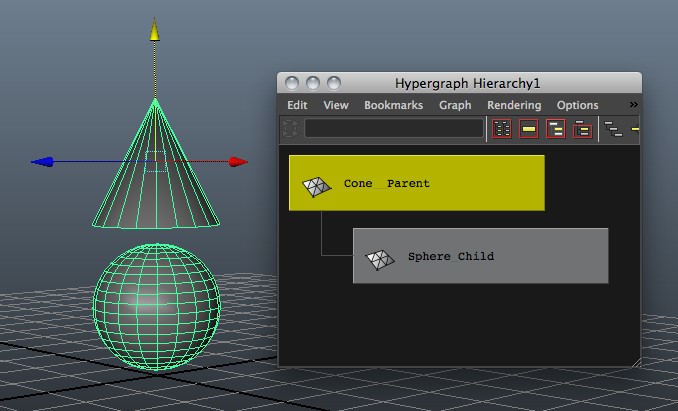


#### HierarchyGrouping



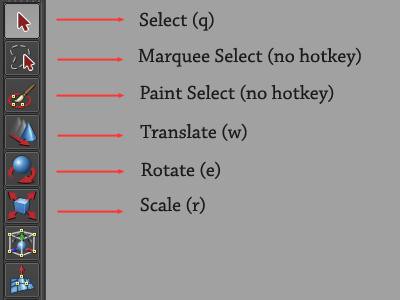
Tocontrolmultipleobjectswithonenode,youcangroupobjectstogetherunderanewtransformnode.Bygroupingobjects,youcanmove,shade,texture,anddomanyotheractionstoalltheobjects at thesametime.

**Parenting**



Thechildobjectshaveindependentnodes,forexampleachildcanrotatebyitselfwithoutrotatingthe parent. However, ifthe parent is rotated or moved the child follows.

#### ObjectManipulationTools



Maya's tool selection icons on the left sideof the userinterface.

#### There aretwo different waysto bringup the translate,scale, androtate tools:

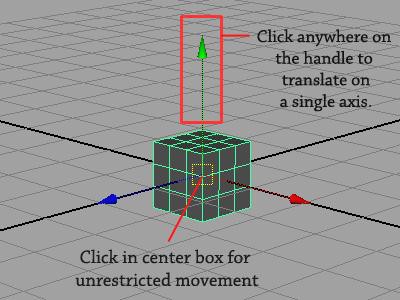
First,theycanbeaccessedfromthetoolboxpanel(picturedabove)ontheleftsideofview-port.

Thesecond(preferredmethod)istousekeyboardhotkeys.

Withanobjectselected,usethefollowinghotkeystoaccessMaya'stranslate,rotate,andscaletools:

Translate -**w**.Rotate-**e**.

Scale -**r**.



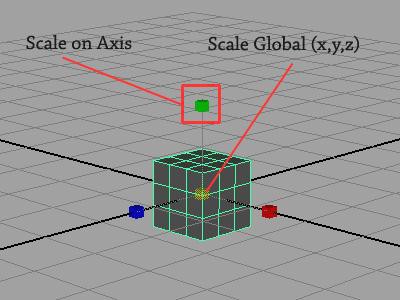
To exit anytool, hit**q** to return to selection mode.**Translate (Move)**

Select the objectyou createdand strikethe**w**keyto bringup the translation tool.

Whenyouaccessthetool,acontrolhandlewillappearatyourobject'scentralpivotpoint,withthree arrowsaimedalong the X, Y, andZaxes.

Tomoveyourobjectaway fromtheorigin,clickanyoneofthearrowsanddragtheobjectalongthataxis.Clickinganywhereonthearroworshaftwillconstrainmovementtotheaxisitrepresents,soifyouonlywanttomoveyourobjectvertically,simplyclickanywhereontheverticalarrowandyourobject will be constrainedto verticalmovement. Ifyou'dliketotranslatetheobjectwithoutconstrainingmotiontoasingleaxis,clickingintheyellowsquareatthecenterofthetooltoallowfreetranslation.Whenmovinganobjectonmultipleaxes,it'softenbeneficialtoswitchintooneofyourorthographiccameras(byclicking**spacebar**)formorecontrol.

**Scale**

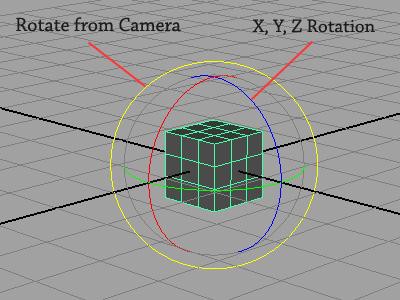


Thescale tool functions almost exactlylikethetranslatetool.

Toscalealonganyaxis,simplyclickanddragthe(red,blue,orgreen)boxthatcorrespondstothe axisyou'd liketo manipulate.

Toscaletheobjectglobally(simultaneouslyonallaxes),clickanddragtheboxlocatedatthecenterof thetool.

#### Rotate



Asyoucan see, the rotation tool appearsand operates slightlydifferent from the translate andscale tools.

Liketranslateandscale,youcanconstrainrotationtoasingleaxisbyclickinganddragginganyofthe three inner rings(red,green, blue) visibleon the tool.

Youcanfreelyrotatetheobjectalongmultipleaxes,bysimplyclickinganddragginginthegapsbetweenrings,howeveryou'reaffordedalotmorecontrolbyrotatinganobjectoneaxisatatime.

Finally,byclickinganddraggingontheouterring(yellow),youcanrotateanobjectperpendicular to the camera.

**ANIMATION**

Mostanimationsystemsusethe*frame*asthebasicunitofmeasurementbecauseeachframeisplayedback in rapid succession to providethe illusion of motion.

Theframerate(framespersecond)thatisusedtoplaybackananimationisbasedonthemedium that the animation will be playedback(for example,film,TV,videogame,etc.)

Whenyousetseveralkeysatdifferenttimeswithdifferentvalues,Mayageneratestheattributevaluesbetween thosetimes as the scene plays back eachframe. The

#### AnimationControl and Interface

WithMaya'sanimationcontrols,youchoosehowtokeyandplayananimation.TwocomponentsofMaya'suserinterfacearespecifictoanimation:theRangeSliderandtheTimeSlider.Youcanalsoquicklyaccessandeditanimationpreferencesfromtheanimationcontrolsarea.



Fig1Animation controls

Between the Range sliderand the AnimationPreferences button arethe currentcharactercontrolfeatures andthe AutoKeybutton.

#### Time Slider

TheTimeSliderisavitalpartoftheanimationinterfaceinMaya.TheTimeSliderControlsthePlaybackrange,keys and breakdownswith in theplaybackrange.

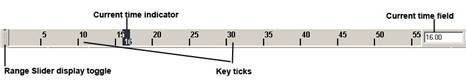


Fig2Time slider

Click in theTime Sliderareaand dragleft andright to"scrub"the animation backand forward intime.

#### KeyTicks

KeyTicksareredmarks in theTimeSliderwhereyouset akeyfortheselectedobject.Breakdownsareaspecialtypeofkey.ThevisibilityofKeyTickscanbeturnedofforoninthePreferencewindow.ThecurrentTimeindicatorisagrayblockontheTimeSlider.Youcandragit to move forwardand backward inyour animation.

#### CurrentTimeField

Ablacklineindicatesthecurrenttimefield.Whenkeyshavebeensetforthecurrentlyselectedobject, thin verticalred lines appear in theTime Sliderarea to indicate thetimes for those keys.

#### Range Slider

TheRangeSlidercontrolstheplaybackrangereflectedintheTimeSlider.TheRangeSlidersetsthetotallengthoftheanimationinframes.YoucanalsousetheRangeSlidertotemporarilylimit the range ofplaybackandset the playback start and endframes.



#### Fig 3 Range slider

* + YoucantogglewhethertheRangeSliderisDisplayedorhiddenby selectingDisplay>UIelements>Range Slider.
  + Animation Start Time sets the start timeof theanimation.
  + Animation End Timesets the endtime ofthe animation.
  + PlaybackStartTime Thisshowsthecurrentstarttimefortheplaybackrange.Youcanchange it byenteringanew start time.
  + PlaybackEndTime Thisshowsthecurrentendtimefortheplaybackrange.Youcanchange it byenteringanew endframe.
  + RangeSliderBar Thisletsyoucontroltheplaybackrangeofyouranimationuptothelimits of the Animationstart/endsettings.

YouusethePreferencesdialogboxtochangevaluesfortheanimationtimelineandplayback.Youcanalsosetthetotaltimeforyouranimation,thesizeofthetimeline,andotherrelatedfeatures.

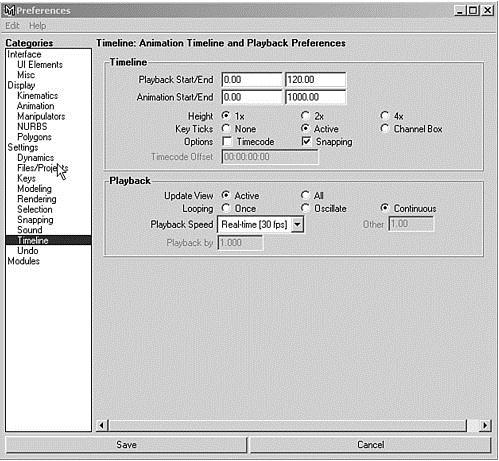


Fig4 Preferences dialogbox

#### Terms inAnimation

**Frame Rate**

Framerateisthefirstaspectofanimation.Bydefault,MayasetsyouranimationtoFilm,whichplays at 24fps.You use30fps in the UnitedStates and 25fps in othercountries.

#### Range

Therangeofananimationdeterminesthetotallengthinframes.Then,multiplytheanimation'slengthinseconds bytheframerate.Fore.ginthiscaseyouare using24fpsandanimationlengthis of 2seconds.

24fpsX 2secs=48Frames

#### SettingKeys

YoucansetaKeybyselectingAnimate>SetKey.Theattributessetbythismenuitemdependon the setKeyoption settings.

#### AnimationTypes

Thereare followingtypes to animateyourscene.

#### PathAnimation

Inthismethod,youcreateaNURBS-basedcurveandthenattachanobjecttoitinyourscene.Theobjectthenfollowsthecurvedpathtosimulatemotion.Youcanchooseatwhichtimetheobjectispositionedatanypointalongthepath,sotheobjectcanreverseitself,pause,oroscillate,ifyouwant.Theobjectautomaticallyrotatesfromsidetosideasthecurvechangesdirections.Iftheobjectisgeometry,itcanalsobeautomaticallydeformedtofollowthecontoursofthe curve.

SelectAnimate>Motionpaths>Attach to Motion path

#### To Animate an object along a surface

* + Choose create >Nurbs Primitives >Planeto createa Nurbsplane.
  + SelectModify>TransformationTools>ProportionalModificationTool to introducecontours on the plane.
  + Select Modify> MakeLive,then drawacurveonthe plane.
  + Createan object to animate alongthe path , and shift-click on the curve onsurface toselect it.
  + SelectAnimate>MotionPaths>AttachtoMotionPathoptionwindow.Ensurethatfollowison,andsettheupdirectiontonormalsothattheobjectwillstaynormaltothesurface.
  + Click playto seeyouranimation.

#### FlowPath ObjectFunction

TheFlowPath Object function creates a lattice around anobject.SelectAnimate>Motionpaths>FlowPath Object

#### Keyframe Animation

Keyframeanimationisthestandardanimationmethod.Inthismethod,yousetkeysforanobject'sextremepositionsandletthecomputerfillinthein-betweenmotion.Akeyisananchorpointforaparticularattributeatadesignatedtime.Whentheanimationreachesthatspecifiedtime,theobject'sattributewillbeatthevalueyouset.Asyousetkeys,youspecifythetimeatwhich those changes in the attribute's valuetake place.

Tosetkeyswiththeautokeyframemethod,youclicktheAutoKeyframebuttonintheRangeSlider(itturnsredtoindicatethatit'senabled).Withautokeyframing,youcananimatequicklybysimplydraggingthe Time Slider to agivenframeand then changing anattribute.

#### Nonlinear Animation

Nonlinearanimationisamoreadvancedmethodofanimation.Unlikekeyframing,nonlinearanimationiscompletelyindependentoftime.Youblendandlayeranimationsequences—calledclips—tosetupthemotionforobjects.Youcanalsousethismethodtoexplorevariationsinpartsoftheanimationwithoutlosingyourpreviousworkoraffectingotherpartsoftheanimation.Forexample-youcanmakethewalkingpartoftheanimationaclipandthenadjustthe legmotion without affectingthe waythe restof thecharactermoves.

#### Graph Editor

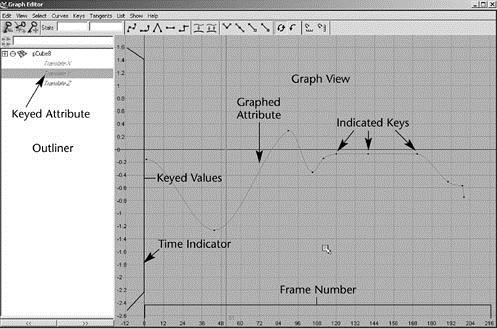
TheGraphEditor is ahelpful tool fortweakingvalues forkeysyou haveset.Itgivesyouavisualrepresentation—acurvedline—oftheattributesthatareanimated.Theanimationtime goesfromlefttoright,andanykeyedvariableappearsasalinethatrampsupordowntoindicateitsvalueovertime.Itcanhelpyouvisualizehowthingsarechangingandhowfast.Youcanpanandzoom this panel like anyother.

Fig5Graph Editor

To useit as a free-floatingwindow, simplyopen itfrom the Hotbox(Window>AnimationEditors>GraphEditor).

#### GraphEditor'sComponentsMenu bar

TheGraphEditormenubarcontainstoolsandoperationsformanipulatinganimationcurvesandkeyswithinthegraphviewoftheGrapheditor.TheEditmenuissimilartotheoneintexteditors or word processors,exceptthatyou're workingwith keysinstead oftext.

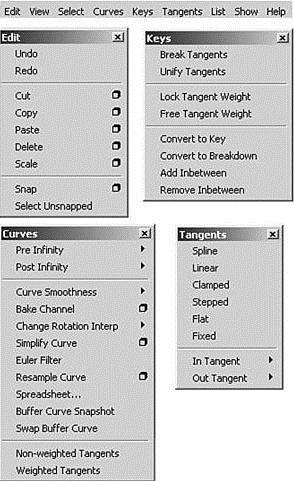


Fig6 Menu Bar

#### TheEdit Menu

Themenu items appear under Edit menu behavein a similarfashion to themain Edit menu in themodelingview.

#### The ViewMenu

This menucontrols which componentsarevisible,andthereforeeditable,in the graph viewofthe GraphEditor.

#### The SelectMenu

These options control which componentof ananimationcurve areavailablefor selectionandediting.

#### The CurvesMenu

TheCurves menugivesyoucontrolover howthecurves areset up with thekeys inyour scene.

#### The Keysmenu

This menuincludes Tangents whichcauses the manipulation of an inorout tangenthandle.

#### The TangentsMenu

This describes the entryand exit of curve segments from a key.

#### TheList Menu

This menuLoads the objects.

#### Toolbar

Thetoolbargivesyou quick access to functions for modifyinganimationcurves and keys.

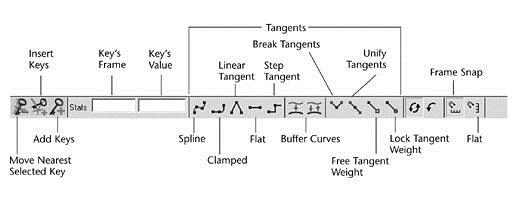


Fig7Toolbar

* + BuffercurvesUseBufferCurvesnapshotandswapBuffercurvestocomparechangestothe current animation curvewith its previousshape.
  + BreakTangentsAllowsmanipulationoftheinandouttangenthandlesindividuallysoyoucan edit the curvesegment enteringthe keywithout affectingits opposite handle.
  + UnifyTangentscausesthemanipulationofaninorouttangenthandletoaffectitsoppositehandleequally.Itretainstherelativepositionofthetangenthandlesevenaftertangentsare individuallyadjusted.
  + LockTangentWeightspecifiesthatwhenyoumoveatangent,onlyitsanglecanbechanged.
  + FreeTangentWeightspecifiesthatwhenyoumoveatangent,onlyitsanglecanbechanged. This allows theweight ofa tangent to beadjustedaswellas theangle.
  + Clampedtangentcreatesananimationcurvethathasthecharacteristicsoflinearandsplinecurves.Thekey'stangentswillbesplineunlessthevalueoftwoadjacentkeysareveryclose.
  + Step tangentcreatesananimation curve whoseout tangent is aflatcurve.
  + FlatSets thein and out tangents of thekeyto be horizontal.

#### TheDope Sheet

TheDopeSheet is anotheranimationeditor in Mayathat is similarto the GraphEditor. Insteadof displayingcurves, theDopeSheetdisplayskeytimes ascoloredrectangles andletsyoueditevent timingin blocks ofkeyframes andsynchronizemotion to asound file.

#### To openthe Dope Sheet

Select Window >Animation Editors >DopeSheet

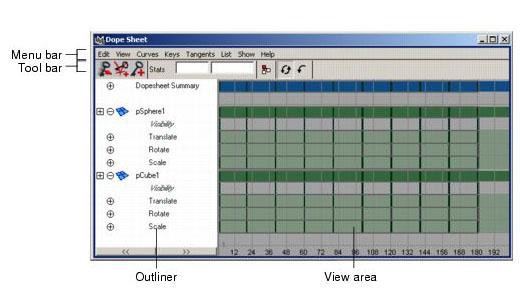


Fig8Dopesheet editor

# VIVA QUESTIONS

QUESTION 1. What is Maya ?

QUESTION 2. Why do we use software such as Adobe Flash or Maya ?

QUESTION 3. What is a ‘timeline’ in Maya ?

QUESTION 4. What are different types of Shapes that we can use in Maya ?

QUESTION 5. Why do we use Nurbs in Maya ?

# EXPERIMENT 8

# AIM

To bounceaball usingAnimation.

# INSTRUCTION

Step1:SelectAnimationfrommainmenubar.

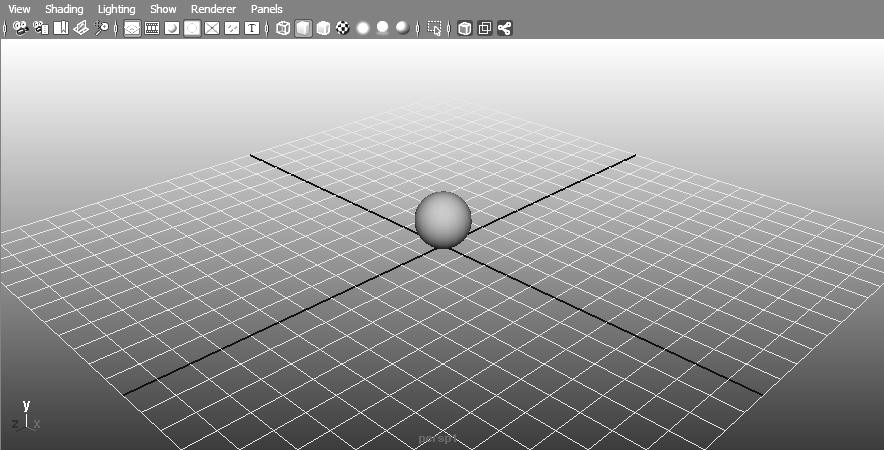
Step2:ClickCreate onMayatitlebarandthenclickonPolygon‘s primitives andthenclickonSphere‘.

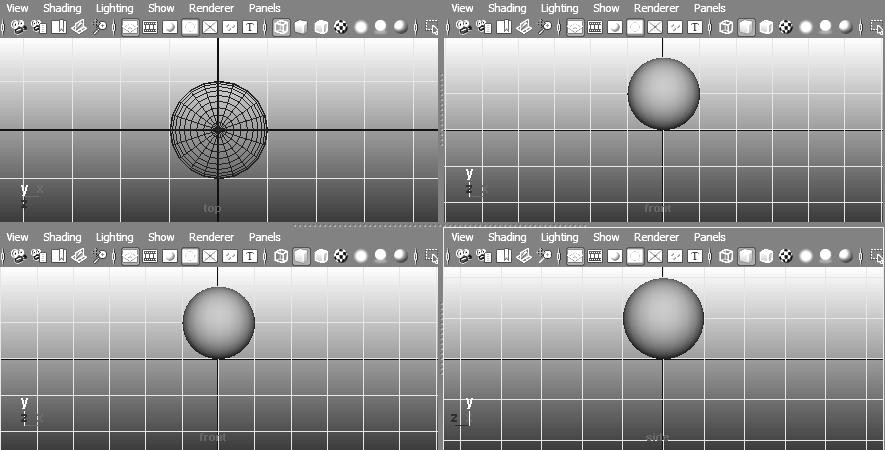
Step3:ClickMovetool tomovetheballtopositioninaframe.

Step4:Fix thepositionoftheballinaframebypressingtheSetkey(Shortcut-S).

Step 5:Move to the nextframeandset another position of the ball usingSetkey.

Step6:ClickPlaytoviewthemovingball.





# VIVA QUESTIONS

QUESTION 1. What do you mean by animation ?

QUESTION 2. List various other software besides Maya that are used for animation.

QUESTION 3. What is a ‘keyframe’ in Maya ?

QUESTION 4. How do you define color on an object in Maya ?

QUESTION 5. What do you mean by stage in Maya ?

# 8. HINTS FOR PROGRAMS BEYOND SYLLABUS

1. Write a programto rotatea circle(alternatively inside andoutside) around thecircumference ofanothercircle (animation).

HINT CODE

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

#include<dos.h>

int xc=50,yc=200,r=35;

int x[15],y[15];

void drawcircles()

{

setcolor(YELLOW);

circle(xc,yc,r);

circle(xc,yc,r+5);

}

void main()

{

double angle=0,theta;

int i,a;

int gd=DETECT,gm;

initgraph(&gd,&gm,"..**\\**bgi");

a=xc+r;

while(!kbhit())

{

while(a<=630)

{

theta=M\_PI\*angle/180;

cleardevice();

drawcircles();

for(i=0;i<18;i++)

{

theta=M\_PI\*angle/180;

x[i]=xc+r\*cos(theta);

y[i]=yc+r\*sin(theta);

angle+=20;

line(xc,yc,x[i],y[i]);

}

angle+=2; xc+=2; a=xc+r;

delay(50);

}

xc=50; r=35; a=xc+r;

}

getch();

closegraph();

}

1. Writeaprogramtodrawacarusinginbuildgraphicsfunctionandtranslateitfrombottom leftcorner to right bottom corner ofscreen (animation).

HINT CODE

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <dos.h>

int main() {

int gd = DETECT, gm;

int i, maxx, midy;

/\* initialize graphic mode \*/

initgraph(&gd, &gm, "X:\\TC\\BGI");

/\* maximum pixel in horizontal axis \*/

maxx = getmaxx();

/\* mid pixel in vertical axis \*/

midy = getmaxy()/2;

for (i=0; i < maxx-150; i=i+5) {

/\* clears screen \*/

cleardevice();

/\* draw a white road \*/

setcolor(WHITE);

line(0, midy + 37, maxx, midy + 37);

/\* Draw Car \*/

setcolor(YELLOW);

setfillstyle(SOLID\_FILL, RED);

line(i, midy + 23, i, midy);

line(i, midy, 40 + i, midy - 20);

line(40 + i, midy - 20, 80 + i, midy - 20);

line(80 + i, midy - 20, 100 + i, midy);

line(100 + i, midy, 120 + i, midy);

line(120 + i, midy, 120 + i, midy + 23);

line(0 + i, midy + 23, 18 + i, midy + 23);

arc(30 + i, midy + 23, 0, 180, 12);

line(42 + i, midy + 23, 78 + i, midy + 23);

arc(90 + i, midy + 23, 0, 180, 12);

line(102 + i, midy + 23, 120 + i, midy + 23);

line(28 + i, midy, 43 + i, midy - 15);

line(43 + i, midy - 15, 57 + i, midy - 15);

line(57 + i, midy - 15, 57 + i, midy);

line(57 + i, midy, 28 + i, midy);

line(62 + i, midy - 15, 77 + i, midy - 15);

line(77 + i, midy - 15, 92 + i, midy);

line(92 + i, midy, 62 + i, midy);

line(62 + i, midy, 62 + i, midy - 15);

floodfill(5 + i, midy + 22, YELLOW);

setcolor(BLUE);

setfillstyle(SOLID\_FILL, DARKGRAY);

/\* Draw Wheels \*/

circle(30 + i, midy + 25, 9);

circle(90 + i, midy + 25, 9);

floodfill(30 + i, midy + 25, BLUE);

floodfill(90 + i, midy + 25, BLUE);

/\* Add delay of 0.1 milli seconds \*/

delay(100);

}

getch();

closegraph();

return 0;

}

1. Writeaprogramtodrawballoonsusingin-builtgraphicsfunctionandtranslateitfrom bottom to top ofscreen (animation).

HINT CODE

#include<stdio.h>  
#include<conio.h>  
#include<graphics.h>  
void main()  
{  
int gd=DETECT,gm;  
initgraph(&gd,&gm,"");  
setbkcolor(WHITE);  
setcolor(DARKGRAY);  
ellipse(200,100,0,360,30,40);  
circle(202,143,3);  
line(203,146,210,258);  
setfillstyle(SOLID\_FILL,LIGHTRED);  
floodfill(200,100,DARKGRAY);  
settextstyle(1,HORIZ\_DIR,4);  
outtextxy(190,84,"A");  
ellipse(250,110,0,360,30,40);  
circle(252,153,3);  
line(253,156,246,240);  
setfillstyle(SOLID\_FILL,YELLOW);  
floodfill(250,110,DARKGRAY);  
outtextxy(240,94,"L");  
ellipse(300,100,0,360,30,40);  
circle(302,143,3);  
line(303,146,310,258);  
setfillstyle(SOLID\_FILL,LIGHTGREEN);  
floodfill(300,100,DARKGRAY);  
outtextxy(290,84,"L");  
ellipse(100,100,0,360,30,40);  
circle(102,143,3);  
line(103,146,110,258);  
setfillstyle(SOLID\_FILL,MAGENTA);  
floodfill(100,100,DARKGRAY);  
outtextxy(90,84,"B");  
ellipse(400,100,0,360,30,40);  
circle(402,143,3);  
line(403,146,410,258);  
setfillstyle(SOLID\_FILL,LIGHTMAGENTA);  
floodfill(400,100,DARKGRAY);  
outtextxy(390,84,"O");  
ellipse(450,110,0,360,30,40);  
circle(452,153,3);  
line(453,146,460,258);  
setfillstyle(SOLID\_FILL,LIGHTCYAN);  
floodfill(450,110,DARKGRAY);  
outtextxy(440,94,"O");  
ellipse(500,100,0,360,30,40);  
circle(502,143,3);  
line(503,146,510,258);  
setfillstyle(SOLID\_FILL,GREEN);  
floodfill(500,100,DARKGRAY);  
outtextxy(490,84,"N");  
setcolor(GREEN);  
settextstyle(4,HORIZ\_DIR,8);  
outtextxy(100,280,"BALLOONS");  
getch();  
closegraph();  
}

1. Writeaprogramtodrawacubeusinginbuildlibraryfunctionandperform3Dtransformations
   1. Translations in x,y,zdirections
   2. Rotation byangle450about zaxis,rotationby600abouty-axis in succession.
   3. Scalingin x-directionbyafactor of2, scaling iny- directionbyafactor of3.

HINT CODE

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <stdlib.h>

#include <dos.h>

#include <conio.h>

#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 main()

{

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, "..**\\**bgi");

while (!kbhit())

rotate();

closegraph();

}

1. Write aprogram to show animation ofa ballmovingin a helical path. (animation)

CODE HINT

#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;

}

1. Write aprogram to show animation of solarsystem. (animation)

CODE HINT

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <dos.h>

#include <math.h>

/\* manipulates the position of planets on the orbit \*/

void planetMotion(int xrad, int yrad, int midx, int midy, int x[60], int y[60]) {

int i, j = 0;

/\* positions of planets in their corresponding orbits \*/

for (i = 360; i > 0; i = i - 6) {

x[j] = midx - (xrad \* cos((i \* 3.14) / 180));

y[j++] = midy - (yrad \* sin((i \* 3.14) / 180));

}

return;

}

int main() {

/\* request auto detection \*/

int gdriver = DETECT, gmode, err;

int i = 0, midx, midy;

int xrad[9], yrad[9], x[9][60], y[9][60];

int pos[9], planet[9], tmp;

/\* initialize graphic mode \*/

initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");

err = graphresult();

if (err != grOk) {

/\* error occurred \*/

printf("Graphics Error: %s",

grapherrormsg(err));

return 0;

}

/\* mid positions at x and y-axis \*/

midx = getmaxx() / 2;

midy = getmaxy() / 2;

/\* manipulating radius of all 9 planets \*/

planet[0] = 7;

for (i = 1; i < 9; i++) {

planet[i] = planet[i - 1] + 1;

}

/\* offset position for the planets on their corresponding orbit \*/

for (i = 0; i < 9; i++) {

pos[i] = i \* 6;

}

/\* orbits for all 9 planets \*/

xrad[0] = 60, yrad[0] = 30;

for (i = 1; i < 9; i++) {

xrad[i] = xrad[i - 1] + 30;

yrad[i] = yrad[i - 1] + 15;

}

/\* positions of planets on their corresponding orbits \*/

for (i = 0; i < 9; i++) {

planetMotion(xrad[i], yrad[i], midx, midy, x[i], y[i]);

}

while (!kbhit()) {

/\* drawing 9 orbits \*/

setcolor(WHITE);

for (i = 0; i < 9; i++) {

ellipse(midx, midy, 0, 360, xrad[i], yrad[i]);

}

/\* sun at the mid of the solar system \*/

setcolor(YELLOW);

setfillstyle(SOLID\_FILL, YELLOW);

circle(midx, midy, 20);

floodfill(midx, midy, YELLOW);

/\* mercury in first orbit \*/

setcolor(CYAN);

setfillstyle(SOLID\_FILL, CYAN);

pieslice(x[0][pos[0]], y[0][pos[0]], 0, 360, planet[0]);

/\* venus in second orbit \*/

setcolor(GREEN);

setfillstyle(SOLID\_FILL, GREEN);

pieslice(x[1][pos[1]], y[1][pos[1]], 0, 360, planet[1]);

/\* earth in third orbit \*/

setcolor(BLUE);

setfillstyle(SOLID\_FILL, BLUE);

pieslice(x[2][pos[2]], y[2][pos[2]], 0, 360, planet[2]);

/\* mars in fourth orbit \*/

setcolor(RED);

setfillstyle(SOLID\_FILL, RED);

pieslice(x[3][pos[3]], y[3][pos[3]], 0, 360, planet[3]);

/\* jupiter in fifth orbit \*/

setcolor(BROWN);

setfillstyle(SOLID\_FILL, BROWN);

pieslice(x[4][pos[4]], y[4][pos[4]], 0, 360, planet[4]);

/\* saturn in sixth orbit \*/

setcolor(LIGHTGRAY);

setfillstyle(SOLID\_FILL, LIGHTGRAY);

pieslice(x[5][pos[5]], y[5][pos[5]], 0, 360, planet[5]);

/\* uranus in sevth orbit \*/

setcolor(BROWN);

setfillstyle(SOLID\_FILL, BROWN);

pieslice(x[6][pos[6]], y[6][pos[6]], 0, 360, planet[6]);

/\* neptune in eigth orbit \*/

setcolor(LIGHTBLUE);

setfillstyle(SOLID\_FILL, LIGHTBLUE);

pieslice(x[7][pos[7]], y[7][pos[7]], 0, 360, planet[7]);

/\* pluto in ninth orbit \*/

setcolor(LIGHTRED);

setfillstyle(SOLID\_FILL, LIGHTRED);

pieslice(x[8][pos[8]], y[8][pos[8]], 0, 360, planet[8]);

/\* checking for one complete rotation \*/

for (i = 0; i < 9; i++) {

if (pos[i] <= 0) {

pos[i] = 59;

} else {

pos[i] = pos[i] - 1;

}

}

/\* sleep for 100 milliseconds \*/

delay(100);

/\* clears graphic screen \*/

cleardevice();

}

/\* deallocate memory allocated for graphic screen \*/

closegraph();

return 0;

}

1. Create a Rainbow using graphics in C Program

CODE HINT

// A C program to make a rainbow. This program would only

// work in Turbo C compiler in DOS compatible machine

#include<stdio.h>

#include<graphics.h>;

#include<dos.h>;

// function for making of rainbow

void rainbow()

{

// auto detection

int gdriver = DETECT,gmode;

int x, y, i;

// initialize graphics mode(passed three arguments to

// initgraph function)

// &gdriver is the address of gdriver variable, &gmode is

// the address of gmode and

// "C:\\Turboc3\\BGI" is the directory path where BGI files are stored

initgraph(&gdriver,&gmode,"C:\\Turboc3\\BGI");

x = getmaxx() / 2;//finding centre x-ordinate of screen

y = getmaxy() / 2;//finding centre y-ordinate of screen

for (i=30; i<200; i++)

{

// delay function under dos.h for holding the

// function for some time

delay(100);

// selecting color for making of rainbow

setcolor(i/10);

// making of arc with fixed centre and increasing radius

arc(x, y, 0, 180, i-10);

}

}

// driver program

int main()

{

rainbow();

return 0;

}

1. Create a Digital clock using graphics in C Program. (animation)

CODE HINTS

#include <conio.h>

#include <graphics.h>

#include <time.h>

#include <dos.h>

#include <string.h>

int main() {

int gd = DETECT, gm;

int midx, midy;

long current\_time;

char timeStr[256];

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

/\* mid pixel in horizontal and vertical axis \*/

midx = getmaxx() / 2;

midy = getmaxy() / 2;

while (!kbhit()) {

cleardevice();

setcolor(WHITE);

setfillstyle(SOLID\_FILL, WHITE);

rectangle(midx - 250, midy - 40, midx + 250, midy + 40);

floodfill(midx, midy, WHITE);

/\* Get Current epoch time in seconds \*/

current\_time = time(NULL);

/\* store the date and time in string \*/

strcpy(timeStr, ctime(&current\_time));

setcolor(RED);

settextjustify(CENTER\_TEXT, CENTER\_TEXT);

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 4);

moveto(midx, midy);

/\* print current time \*/

outtext(timeStr);

/\* Add delay of 1000 milliseconds(1 second) \*/

delay(1000);

}

getch();

closegraph();

return 0;

}

1. Design a C Program for Tic Tac Toe Game. (game)

CODE HINT

// A C++ Program to play tic-tac-toe

#include<bits/stdc++.h>

using namespace std;

#define COMPUTER 1

#define HUMAN 2

#define SIDE 3 // Length of the board

// Computer will move with 'O'

// and human with 'X'

#define COMPUTERMOVE 'O'

#define HUMANMOVE 'X'

// A function to show the current board status

void showBoard(char board[][SIDE])

{

printf("\n\n");

printf("\t\t\t %c | %c | %c \n", board[0][0],

board[0][1], board[0][2]);

printf("\t\t\t--------------\n");

printf("\t\t\t %c | %c | %c \n", board[1][0],

board[1][1], board[1][2]);

printf("\t\t\t--------------\n");

printf("\t\t\t %c | %c | %c \n\n", board[2][0],

board[2][1], board[2][2]);

return;

}

// A function to show the instructions

void showInstructions()

{

printf("\t\t\t Tic-Tac-Toe\n\n");

printf("Choose a cell numbered from 1 to 9 as below"

" and play\n\n");

printf("\t\t\t 1 | 2 | 3 \n");

printf("\t\t\t--------------\n");

printf("\t\t\t 4 | 5 | 6 \n");

printf("\t\t\t--------------\n");

printf("\t\t\t 7 | 8 | 9 \n\n");

printf("-\t-\t-\t-\t-\t-\t-\t-\t-\t-\n\n");

return;

}

// A function to initialise the game

void initialise(char board[][SIDE], int moves[])

{

// Initiate the random number generator so that

// the same configuration doesn't arises

srand(time(NULL));

// Initially the board is empty

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

{

for (int j=0; j<SIDE; j++)

board[i][j] = ' ';

}

// Fill the moves with numbers

for (int i=0; i<SIDE\*SIDE; i++)

moves[i] = i;

// randomise the moves

random\_shuffle(moves, moves + SIDE\*SIDE);

return;

}

// A function to declare the winner of the game

void declareWinner(int whoseTurn)

{

if (whoseTurn == COMPUTER)

printf("COMPUTER has won\n");

else

printf("HUMAN has won\n");

return;

}

// A function that returns true if any of the row

// is crossed with the same player's move

bool rowCrossed(char board[][SIDE])

{

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

{

if (board[i][0] == board[i][1] &&

board[i][1] == board[i][2] &&

board[i][0] != ' ')

return (true);

}

return(false);

}

// A function that returns true if any of the column

// is crossed with the same player's move

bool columnCrossed(char board[][SIDE])

{

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

{

if (board[0][i] == board[1][i] &&

board[1][i] == board[2][i] &&

board[0][i] != ' ')

return (true);

}

return(false);

}

// A function that returns true if any of the diagonal

// is crossed with the same player's move

bool diagonalCrossed(char board[][SIDE])

{

if (board[0][0] == board[1][1] &&

board[1][1] == board[2][2] &&

board[0][0] != ' ')

return(true);

if (board[0][2] == board[1][1] &&

board[1][1] == board[2][0] &&

board[0][2] != ' ')

return(true);

return(false);

}

// A function that returns true if the game is over

// else it returns a false

bool gameOver(char board[][SIDE])

{

return(rowCrossed(board) || columnCrossed(board)

|| diagonalCrossed(board) );

}

// A function to play Tic-Tac-Toe

void playTicTacToe(int whoseTurn)

{

// A 3\*3 Tic-Tac-Toe board for playing

char board[SIDE][SIDE];

int moves[SIDE\*SIDE];

// Initialise the game

initialise(board, moves);

// Show the instructions before playing

showInstructions();

int moveIndex = 0, x, y;

// Keep playing till the game is over or it is a draw

while (gameOver(board) == false &&

moveIndex != SIDE\*SIDE)

{

if (whoseTurn == COMPUTER)

{

x = moves[moveIndex] / SIDE;

y = moves[moveIndex] % SIDE;

board[x][y] = COMPUTERMOVE;

printf("COMPUTER has put a %c in cell %d\n",

COMPUTERMOVE, moves[moveIndex]+1);

showBoard(board);

moveIndex ++;

whoseTurn = HUMAN;

}

else if (whoseTurn == HUMAN)

{

x = moves[moveIndex] / SIDE;

y = moves[moveIndex] % SIDE;

board[x][y] = HUMANMOVE;

printf ("HUMAN has put a %c in cell %d\n",

HUMANMOVE, moves[moveIndex]+1);

showBoard(board);

moveIndex ++;

whoseTurn = COMPUTER;

}

}

// If the game has drawn

if (gameOver(board) == false &&

moveIndex == SIDE \* SIDE)

printf("It's a draw\n");

else

{

// Toggling the user to declare the actual

// winner

if (whoseTurn == COMPUTER)

whoseTurn = HUMAN;

else if (whoseTurn == HUMAN)

whoseTurn = COMPUTER;

// Declare the winner

declareWinner(whoseTurn);

}

return;

}

// Driver program

int main()

{

// Let us play the game with COMPUTER starting first

playTicTacToe(COMPUTER);

return (0);

}

# 9. SAMPLE VIVA QUESTIONS

1. What is scan conversion?

A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is called scan conversion

1. Write the properties of video display devices?

Properties of video display devices are persistence, resolution, and aspect ratio

1. What is rasterization?

The process of determining the appropriate pixels for representing picture or graphics object is known as rasterization

1. Define Computer graphics.

Computer graphics remains one of the most existing and rapidly growing computer fields. Computer graphics maybe defined as a pictorial representation or graphical representation of objects in a computer.

1. Name any four input devices

Four input devices are keyboard, mouse, image scanners, and trackball.

1. Write the two techniques for producing color displays with a CRT?

Beam penetration method, shadow mask method

1. What is vertical retrace of the electron beam?

In raster scan display, at the end of one frame, the electron beam returns to the left top corner of the screen to start the next frame

1. Short notes on video controller?

Video controller is used to control the operation of the display device. A fixed area of the system is reserved for the frame buffer, and the video controller is given direct access to the frame buffer memory

1. What is bitmap?

Some system has only one bit per pixel; the frame buffer is often referred to as bitmap.

1. Differentiate plasma panel display and thin film electroluminescent display?

In plasma panel display, the region between two glass plates is filled with neon gas. In thin film electroluminescent display, the region between two glasses plates are filled with phosphor, such as zinc sulphide doped with manganese.

1. What is resolution?

The maximum number of points that can be displayed without overlap on a CRT is referred to as the resolution.

1. What is horizontal retrace of the electron beam?

In raster scan display, the electron beam return to the left of the screen after refreshing each scan line, is called horizontal retrace of the electron beam.

1. What is filament?

In the CRT, heat is applied to the cathode by directing a current through a coil of wire, is called filament

1. What is pixmap?

Some system has multiple bits per pixel, the frame buffer is often referred to as pixmap.

1. Write the types of clipping?

Point clipping, line clipping, area clipping, text clipping and curve clipping.

1. What is meant by scan code?

When a key is pressed on the keyboard, the keyboard controller places a code carry to the key pressed into a part of the memory called as the keyboard buffer. This code is called as the scan code.

1. List out the merits and demerits of Penetration techniques?

The merits and demerits of the Penetration techniques areas follows. It is an inexpensive technique. It has only four colors. The quality of the picture is not good when it is compared to other techniques. It can display color scans in monitors. Poor limitation etc.

1. What do you mean by emissive and non-emissive displays?

The emissive display converts electrical energy into light energy. The plasma panels, thin film electro-luminescent displays are the examples. The Non-emissive are optical effects to convert the sun light or light from any other source to graphic form. Liquid crystal display is an example

1. List out the merits and demerits of Plasma panel display?

Merits. Refreshing is not required. Produce a very steady image free of Flicker. Less bulky than a CRT. Demerits. Poor resolution of up to 60 d.p.i. It requires complex addressing and wiring. It is costlier than CRT.

1. What is persistence?

The time it takes the emitted light from the screen to decay one tenth of its original intensity is called as persistence.

1. What is Aspect ratio?

The ratio of vertical points to the horizontal points necessary to produce length of lines in both directions of the screen is called the Aspect ratio. Usually the aspect ratio is ¾.

1. What is the difference between impact and non-impact printers?

Impact printer press formed character faces against an inked ribbon on to the paper. A line printer and dot-matrix printer are examples .Non-impact printer and plotters use Laser techniques, inkjet sprays, Xerographic process, electrostatic methods and electro thermal methods to get images onto the papers. Examples are: Inkjet/Laser printers.

1. Define pixel?

Pixel is shortened forms of picture element. Each screen point is referred to as pixel or pel.

1. What is frame buffer?

Picture definition is stored in a memory area called frame buffer or refresh buffer.

1. Where the video controller is used?

A special purpose processor, which is used to control the operation of the display device, is known as video controller or display controller.

1. What is run length encoding?

Run length encoding is a compression technique used to store the intensity values in the frame buffer, which stores each scan line as a set of integer pairs. One number each pair indicates an intensity value, and second number specifies the number of adjacent pixels on the scan line that are to have that intensity value.

1. What is point in the computer graphics system?

The point is a most basic graphical element & is completely defined by a pair of user coordinates (x, y).

1. Write short notes on lines?

A line is of infinite extent can be defined by an angle of slope q and one point on the line P=P(x,y). This can also be defined as y=mx+C where C is the Y intercept.

1. Define Circle?

Circle is defined by its center xc, yc and its radius in user coordinate units. The equation of the circle is (x-xc) + (yyc)= r2.

1. What are the various attributes of a line?

The line type, width and color are the attributes of the line. The line type include solid line, dashed lines, and dotted lines.

1. What is antialiasing?

The process of adjusting intensities of the pixels along the line to minimize the effect of aliasing is called antialiasing.

1. What is Transformation?

Transformation is the process of introducing changes in the shape size and orientation of the object using scaling rotation reflection shearing & translation etc.

1. What is translation?

Translation is the process of changing the position of an object in a straight-line path from one coordinate location to another. Every point (x , y) in the object must under go a displacement to (x|,y|). the transformation is:x| = x + tx ; y| = y+ty

1. What is rotation?

A 2-D rotation is done by repositioning the coordinates along a circular path, in the x-y plane by making an angle with the axes. The transformation is given by:X| = r cos (q + f) and Y| = r sin (q + f).

1. What is scaling?

A 2-D rotation is done by repositioning the coordinates along a circular path, in the x-y plane by making an angle with the axes. The transformation is given by: X| = r cos (q + f) and Y| = r sin (q + f).

1. What is shearing?

The shearing transformation actually slants the object along the X direction or the Y direction as required. ie; this transformation slants the shape of an object along a required plane.

1. What is reflection?

The reflection is actually the transformation that produces a mirror image of an object. For this use some angles and lines of reflection.

1. What are the two classifications of shear transformation?

X shear, y shear

# 10. SOME MORE QUESTIONS

1. Explain the terms pixel, frame & Buffer.
2. What do you mean by Computer Graphics?
3. What are the applications of Computer Graphics?
4. Explain the fundamentals of Raster Graphics.
5. What do you mean by Display Devices?
6. Explain Raster scan system in detail.
7. Explain Random scan system in detail.
8. Differentiate between Raster scan & Random scan system.
9. Explain color CRT Monitor.
10. Explain Plasma panel along with labeled diagram.
11. Write down the algorithm for Line Generation.
12. Write down the algorithm for Circle Generation.
13. Write down the algorithm for Polygon Generation.
14. Write down the algorithm for Polygon filling.
15. What do you mean by Anti-aliasing?
16. Explain 2-D Transformation.
17. Describe Translation, Scaling, Rotation, Reflection.
18. Explain homogeneous Co-ordinates.
19. What do you mean by Graphics primitive?
20. Differentiate boundary fill & flood fill algorithm
21. Explain 3-D Transformation.
22. Describe scaling, translation & rotation.
23. Define Clipping.
24. Discuss Windowing & Clipping Window.
25. Define View port.
26. Explain Line Clipping.
27. Describe Window & View port Transformation.
28. What do you mean by Multimedia- text?
29. What do you mean by Multimedia- fonts?
30. What do you mean by Multimedia- animating text?
31. Explain hyper sound text.
32. What do you mean by MIDI?
33. What do you mean by audio file format?
34. Explain the editing of audio.
35. Explain MCI Image & Bitmaps.
36. Explain the concept of 3-D Modeling.
37. Explain image file formats.
38. Discuss BMP & JPG File formats.
39. Explain animations.
40. Write down the principle of animation.