



# VIT

---

## UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

Vellore - 632 014, Tamil Nadu, India

### **School of Information Technology and Engineering (SITE)**

#### ***Multimedia and Graphics Laboratory Manual***

*[Common for B.Tech. (IT) & (MCA)]*

### ***Index-1***

<b>SINo.</b>	<b>Contents</b>	<b>PageNo.</b>
1	Aim and objectives	4
	<b><u>Multimedia-Flash</u></b>	
2	Introduction	4
3	Working with Layers in Flash	6
4	Tweening in Flash	6
5	Masking in Flash	10
6	Action scripts using buttons	11
7	Basic Tools for Adobe Professional Flash CS5	11
8	Case Study-1	18
	<b><u>Multimedia-Photoshop</u></b>	
9	Basic Tools for Adobe Photoshop Professional CS5	21
10	Case Study-2	27

## ***Index-2***

<b>SINo.</b>	<b>Contents</b>	<b>PageNo.</b>
	<b><u>Computer Graphics</u></b>	
1	How do we use Borland Graphics Interface (graphics.h)?	31
2	Test program	34
3	Graphics Algorithms	35
3.1	Line Drawing Algorithms	35
3.2	Circle Drawing Algorithms	37
3.3	Polygon Filling Algorithms	41
3.4	Basic Transformation methods	44
3.5	Rotation about an arbitrary point	46
3.6	Line Clipping Algorithms	48
3.7	Polygon Clipping Algorithm	51
3.8	Curves Generation	52
3.9	References	54

## 1. Aim and Objectives

The course meets the following student outcomes:

- ✓ An ability to apply knowledge of mathematics, science, computing and information technology appropriate to the discipline.
- ✓ An ability to analyze a problem, and identify and define the computing and technology requirements appropriate to its solution.
- ✓ An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs.
- ✓ An ability to use current techniques, skills, and tools necessary for computing practice
- ✓ An ability to use and apply current technical concepts and practices in the core information technologies
- ✓ An ability to identify and analyze user needs and take them into account in the selection, creation, evaluation and administration of computer-based systems
- ✓ An ability to effectively integrate IT-based solutions into the user environment

## **Multimedia-Flash**

### 2. Introduction

#### Frames and key frames

**Frames** like films, Adobe® Flash® Professional CS5 documents divide lengths of time into frames. In the Timeline, you work with these frames to organize and control the content of your document. You place frames in the Timeline in the order you want the objects in the frames to appear in your finished content.

A **keyframe** is a frame where a new symbol instance appears in the Timeline.

A **keyframe** can also be a frame that includes ActionScript® code to control some aspect of your document.

You can also add a *blank keyframe* to the Timeline as a placeholder for symbols you plan to add later or to explicitly leave the frame blank.

A *property keyframe* is a frame in which you define a change to an object's properties for an animation. FlashPro can *tween*, or automatically fill in, the property values between the property keyframes in order to produce fluid animations. Because property keyframes let you produce animation without drawing each individual frame, they make creating animation easier.

A series of frames containing tweened animation is called a ***motion tween***.

A ***tweened frame*** is any frame that is part of a motion tween. A *static frame* is any frame that is not part of a motion tween. You arrange keyframes and property keyframes in the Timeline to control the sequence of events in your document and its animation.

**Layers** help you organize the artwork in your document. You can draw and edit objects on one layer without affecting objects on another layer. In areas of the Stage with nothing on a layer, you can see through it to the layers below.

There are five types of layers you can use in Flash:

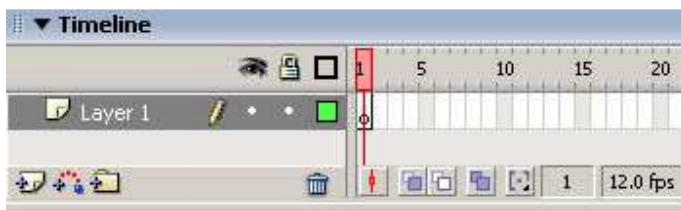
- ✓ **Normal layers** contain most of the artwork in a FLA file.
- ✓ **Mask layers** contain objects used as masks to hide selected portions of layers below them. Masked layers are layers beneath a mask layer that you associate with the mask layer. Only the portion of the masked layer revealed by the mask is visible.
- ✓ **Guide layers** contain strokes that can be used to guide the arrangement of objects on other layers or the motion of classic tween animations on other layers. Guided layers are layers associated with a guide layer. The objects on the guided layer can be arranged or animated along the strokes on the guide layer. Guided layers can contain static artwork and classic tweens, but not motion tweens.
- ✓ **Motion Tween layers** contain objects animated with motion tweens
- ✓ **Armature layers** contain objects with inverse kinematics bones attached.

**Note:** Normal, Mask, Masked, and Guide layers can contain motion tweens or inverse kinematic bones. When these items are present in one of these layers, there are limitations to the types of content that can be added to the layer.

### 3. Working with Layers in Flash

Steps to follow:

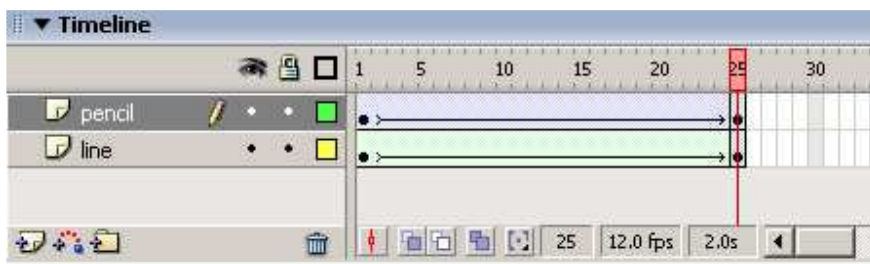
- ✓ Open a new flash file (Ctrl+N). New Document window will appear.
- ✓ Select General panel and choose Type: Flash Document. Press OK.
- ✓ If your timeline window is not open, press (Ctrl+Alt+T).
- ✓ Now you can see a single Layer called "Layer1" in your timeline Window.



- ✓ Create a Shape Tween on Layer1. Similar to the one in Shape Tween Tutorial.
- ✓ Single click on add new layer button.



- ✓ A new layer gets added. By default it will be named "Layer 2".
- ✓ Create a Motion Tween on Layer 2. Similar to the one in Motion Tween Tutorial. After creating two layers, your timeline will look something like the one shown below.



- ✓ Now press (Ctrl+Enter) to view your motion tween.

### 4. Flash can create two types of tweened animation using timeline:

Creation of Motion/Shape tween using timeline is the basics of Flash.

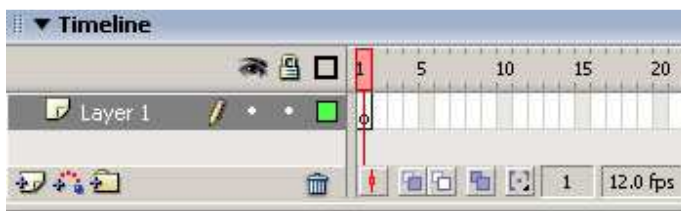
- a. Motion Tween
- b. Shape Tween

Motion tween is nothing but tweening a Symbol's movement from one position to another. To implement Motion Tween all that you have to do is, provide Flash with Symbol's initial position and the end position. Rest is taken care by Flash. Isn't it really simple? For example in the above demonstration, I have converted the pencil object to a Symbol. Provided Flash with its initial and the end position, intermediate tweening is done by Flash.

#### a. Motion Tween in Flash

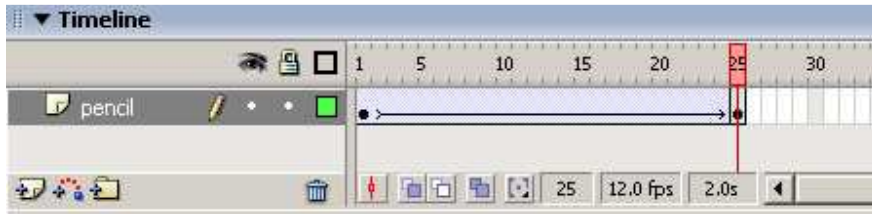
##### Steps to follow:

- ✓ Open a new flash file (Ctrl+N). New Document window will appear. Select General panel and choose Type: Flash Document. Press OK.
- ✓ If your timeline window is not open, press (Ctrl+Alt+T).
- ✓ Now you can see a single Layer called "Layer1" in your timeline Window.



- ✓ Select the first frame. Import your image onto stage, upon which you would want to implement motion tween.  
File>Import>Import to Stage, or just press (Ctrl+R).  
Or you can even draw your own object; you can either choose Rectangular tool or Oval tool from the tool box or draw your desired shape.
- ✓ Now select your object on the stage and press F8 to convert this image to a Symbol. Convert to Symbol window will pop-up.  
Name your Symbol whatever you like. Select **Graphic** behavior and press OK.  
**Note:** You can create motion tween only on symbols. So any object upon which you would want to implement motion tween, First convert the object to a Symbol.
- ✓ Right now your Symbol is in frame1 of Layer1. Select frame 20 and press F6 to insert a new keyframe.

- ✓ Still keeping playhead on frame 20, move your Symbol to any other position other than the present one.
- ✓ Select any frame between, 2 to 19 and select Motion from the tween pop-up menu in the Property inspector. Now your Layer will look something like the one shown below.



- ✓ Now press (Ctrl+Enter) to view your motion tween.

### **What is Motion Guide?**

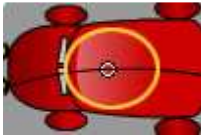
Motion Guide is nothing but moving your symbol in a predefined path such as curves or circles. Learn how to move Flash objects in circular, zig zag or curved paths using Flash motion guide. Download .fla is included at the end of the tutorial.

### **b. Tweening Using Motion Guide**

#### **Steps to follow:**

- ✓ Create a graphic symbol or drag a pre-existing graphic symbol from library onto the stage. Name the layer as "graphic"
- ✓ Right click on the "graphic" label and select "Add Motion Guide" from the pop-up window.
- ✓ A new layer will appear on top of the "graphic" layer with the label "Guide:graphic" along with the guide icon.
- ✓ Draw the path for your symbol in this new layer using pencil or line tool. For example: I drew a circle for my car.
- ✓ Select frame 50 of guide layer and press "F5" to insert frames.
- ✓ Now go to "Frame 1" of "graphic" layer and drag your symbol to one end of your path. While dragging, you will see a bubble on the symbol. That bubble should go right below the path. Something like the one shown below.





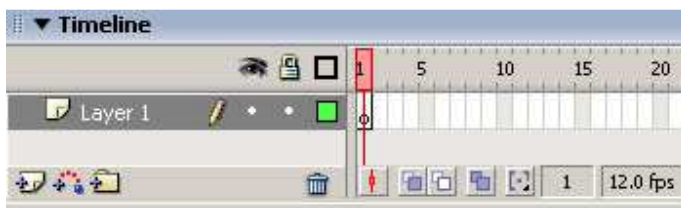
- ✓ Now go to "Frame 50" of "graphic" layer and press F6 to insert a new keyframe.
- ✓ Now drag your symbol to other end of your path. Again, the bubble should go right below the path.
- ✓ Select any frame between 1 and 50 of your "graphic" layer. Right click and select "motion tween" from the pop-up menu.
- ✓ That's it, you are through. Press Ctrl+Enter to view your work.

### c. Shape Tween in Flash

Creation of Motion/Shape tween using timeline is the basics of Flash. By tweening shapes, you can create an effect similar to morphing, making one shape appear to change into another shape over time. Flash can also tween the location, size, and color of shapes.

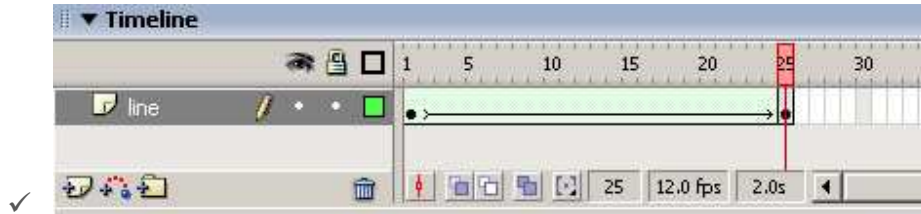
#### Steps to follow:

- ✓ Open a new flash file (Ctrl+N). New Document window will appear. Select **General** panel and choose Type: Flash Document. Press OK.
- ✓ If your timeline window is not open, press (Ctrl+Alt+T).
- ✓ Now you can see a single Layer called "Layer1" in your timeline Window.



- ✓ Select the first frame. Now go to your working area and draw any object. To start off with, maybe you can draw a circle. This is going to be your initial object.  
In the above demonstration, my initial object is a short line.
- ✓ Select frame 20 and press F6 to insert a new keyframe.
- ✓ Still keeping play head on frame 20, delete the object present in your working area. Now draw a different object, maybe a square.  
In the above demonstration, I have drawn a long line.

- ✓ Select any frame between, 2 to 19 and select Shape from the tween pop-up menu in the Property inspector. Now your Layer will look something like the one shown below.



- ✓ Now press (Ctrl+Enter) to view your motion tween.

## 5. Masking in Flash

Masking is revealing portion of your picture or graphic in the layer below. While surfing through net you might have come across lots of beautiful Flash effects such as ripple effect, some wording with sky background or glitter bordering an object, and wondered "How? What is the logic behind this". The answer for all this is masking. This tutorial will teach you the basics of masking in Flash MX 2004. The download .fla file is also included at the end of the tutorial.

### Steps to follow

#### Inserting Layers and naming them

- ✓ By default you will have a layer in your timeline window. Insert one more layer; totally you need two layers to mask an object.
- ✓ Rename the top layer to "Mask" and the layer below that to "background".

#### Creating Shape Tween

- ✓ Import your picture to the "background" layer.
- ✓ Using Oval tool from your tool box, draw a circle in your "Mask" layer and delete it's border.
- ✓ Drag the circle to one end of your picture.
- ✓ Now go to "frame 40" of your "Mask" layer and press "F6" to insert a new keyframe.
- ✓ Now go to "frame 40" of your "background" layer and press "F5" to insert frames, so that your background image is available all through your mask.

- ✓ Select "frame 40" of your "Mask" layer that is your new keyframe, keeping the play head on "frame 40" of "Mask" layer, drag the circle to other end of your picture.
- ✓ Now go back to "frame 1" of your "Mask" layer, keeping the play head on "frame 1" of your "Mask" layer, select Shape tween in your properties window.

### **Masking**

- ✓ Right click on the "Mask" layer (the area where you named the layer not where the frames exist) and select Mask. Your Mask is all ready. Press Ctrl+Enter to view your Mask.

## **6. Action scripts using buttons**

To create a button and give control to it and control button animation

### **Steps to follow:**

- ✓ Open a macromedia flash from the start menu
- ✓ From the toolbar select an object and place it on the work area
- ✓ Create any meaningful object in default layer at the time line
- ✓ Create a motion tweening for that created an object
- ✓ Now go to windows menu and choose button1 from the common library submenu
- ✓ Now on right clicking the button1 and select action then we set the movie control to play (button1)
- ✓ Now create another button2 on right clicking the button2 and select action then we set the movie control to stop (button2)
- ✓ Press ctrl key + enter key to test the movie.

## **7. Basic Tools for Adobe Flash Professional CS5**

There are many basic drawing tools available in Flash. Each is detailed below. Also note that many of these tools have additional options or modes that can be changed in the Options area of the Tools panel.



### **Selection tool:**

- ☐ The first important tool is the Selection tool. This tool is used to make selections on objects and graphics on the drawing area, or stage.
- ☐ You can either click on individual objects to select them, or click and drag to make a rectangular region and select all the objects in it.
- ☐ Once you make a selection, you may perform subsequent actions that affect your selection, such as moving it around the stage, deleting, or altering it in the Properties panel. Hold down the SHIFT key to select multiple individual objects.

### **Sub-selection tool:**

- ☐ The Sub-selection tool is used for selecting and modifying anchor points on curves and lines.
- ☐ Clicking once on a line or curve with the Sub-selection tool reveals the anchor points.

- Anchor points are represented either by a hollow square (a corner point), or a hollow circle (a curve point).
- Clicking on an anchor point with the Sub-selection tool will select that anchor point (then represented by a filled square or circle).
- You can then click and drag the anchor point to move it, ALT-click and drag on a corner point to convert it to a curve point (and thus reveal anchor point tangent handles), or ALT-click on a curve point to convert it to a corner point (and thus remove the tangent handles).

#### **Line tool:**



- The Line tool is an important drawing tool. It functions like the line tool in other drawing programs.
- To use it, click on the stage, drag, and release to draw a straight line.
- Stroke width, style, and color can be changed in the Properties panel.
- Hold down the SHIFT key while dragging to constrain the line angle to increments of 45°.

#### **Lasso tool:**




- The Lasso tool can be used to select objects on the stage.
- It allows creating a freeform selection area by clicking and dragging around an area.
- When you release the mouse button, Flash automatically completes the tool with a straight line.

#### **Pen tool:**





- The Pen tool is used to create precise paths that are either straight lines or smooth curves.
- Stroke width, style, and color, and fill color (for closed paths) can be changed in the Properties panel.
- To use the Pen tool to create straight lines, click anywhere on the stage to define the first anchor point, click again where you want the first segment of the straight line to end (hold down the SHIFT key to constrain the line to increments of 45°), and continue clicking to create more straight line segments.

-  Double click the last point to end the line segments (and create an open curve), or click on the first point you created to close the curve.
-  To create curved lines, follow the same procedure, but instead of simply clicking to create each point, click and drag in the direction you want to curve to go.






#### **Text tool:**


-  The Text tool allows you to draw text on the stage.
-  Position the cursor on the stage where you want to begin your text and then click and start typing.
-  The text font, size, color and paragraph formatting can be changed in the Properties panel.

#### **Oval tool:**






-  The Oval tool allows you to create ovals of any shape and size (including circles).
-  Stroke width, style, and color, and fill color for ovals can be changed in the Properties panel.
-  To create an oval, simply click and drag across the stage to create the oval as you want it.
-  Hold down the SHIFT key while dragging to constrain the oval to a circle.

#### **Rectangle and PolyStar tools:**






-  The Rectangle and PolyStar tools allow you to create rectangles and polygons.
-  To switch between the Rectangle and PolyStar tools, click and hold the tool icon on the Tools panel, then select the desired tool from the menu that appears.
-  Stroke width, style, and color, and fill color for rectangles and polygons can be changed in the Properties panel. In addition, for the PolyStar tool, the number of sides of a polygon, or points on a star, can be changed by clicking on Options in the Properties panel.
-  To create a rectangle or polygon/star, simply click and drag across the stage to create the shape as you want it.
-  While using the Rectangle tool, hold down the SHIFT key while dragging to constrain the rectangle to a square.

-  While using the PolyStar tool, hold down the SHIFT key to constrain the orientation of the polygon or star to increments of 45°.





#### **Pencil tool:**

-  The Pencil tool is used to create lines and shapes on the stage in much the same way you would use a real pencil.
-  Stroke width, style, and color can be changed in the Properties panel.
-  To use the Pencil tool, click and drag across the stage to create a line.
-  Note that you can end your drawing at the same place you started, but the shape you create is not filled in.
-  Hold the SHIFT key while dragging to constrain the line to a horizontal or vertical direction.




#### **Brush tool:**




-  The Brush tool is used much like a paint brush.
-  Click and drag across the stage to paint.
-  The fill color can be changed in the Properties panel.
-  Note that no stroke is produced when using the brush tool.
-  Under the Options section of the Tools panel, you can change the brush size and shape.

#### **Free Transform tool:**



-  The Free Transform tool can be used to transform objects in a variety of ways. To use the tool, click on an object on the stage to reveal transform handles.
-  Then, under the Options section of the Tools panel, choose the type of transformation you want to perform.
-  Options include Rotate and Skew, Scale, Distort, and Envelope.
-  Hold down the SHIFT key while rotating to constrain the angle to increments of 45°.

#### **Fill Transform tool:**




-  The Fill Transform tool allows you transform gradient and bitmap fills.
-  To use it, click on an object that has a gradient or bitmap fill to reveal transform handles.
-  Click and drag the circle handle in the center of the fill to move the gradient or bitmap.

-  To change the width or height of a bitmap fill, or the scale of a gradient fill, click and drag the square handle along the edge of the fill bounding box.
-  To rotate the fill, click and drag the circle handle on the corner of the bounding box.
-  To change the radius of a circular gradient fill, click and drag the middle circle handle on the bounding circle of the fill.






#### **Ink Bottle tool:**

-  The Ink Bottle tool allows you to change the stroke color, width, and style of lines and shape outlines.
-  To use it select the stroke width, style, and color you want to apply in the Properties panel, and click on one or more lines or shapes on the stage to apply the stroke properties to them.


#### **Paint Bucket tool:**

-  The Paint Bucket tool allows you to fill enclosed areas with color.
-  To use it, select the solid color or gradient you want to apply in the Properties panel, and click on one or more enclosed shapes on the stage to fill them with color.
-  Note that under the Options section of the Tools panel, you can choose how you want Flash to handle partially enclosed areas.


#### **Eyedropper tool:**


-  The Eyedropper tool allows you to copy stroke and fill properties from one object to another.
-  To use it, click on a stroke or fill of the object whose properties you want to copy.
-  If you click on the fill, the Eyedropper tool automatically changes to the Paint Bucket tool, with fill properties set the same as the object you clicked on.
-  If you click on the stroke, the Eyedropper tool automatically changes to the Ink Bottle tool, with stroke properties set the same as the object you clicked on.
-  Now click on one or more objects on the stage to apply the stroke or fill properties.

#### **Eraser tool:**


-  The Eraser tool allows you to erase objects on the stage.




 To quickly erase everything on the stage, double click the Eraser tool icon on the Tools panel.


 Note that you can change erasing options in the Options section of the Tools panel.


#### **Hand tool:**

 The Hand tool is used for moving the view of the stage.

 It is especially useful when you have used the Zoom tool to magnify the stage.


 To use it, simply click and drag the stage in the direction you want it to move.

 To temporarily switch between another tool and the Hand tool, hold down the spacebar and click the Hand tool in the Tools panel.

 When done dragging, Flash will return to the tool you were using.

#### **Zoom tool:**

 The Zoom tool is used for magnifying or reducing the view of the stage.

 To use it, click anywhere on the stage to zoom in by a factor of two. Alternatively, you can click and drag to zoom into a region of the stage. Hold down the ALT key while clicking to zoom out by a factor of two.

#### **Drawing and view tools:**



- ✓ In addition to the drawing and view tools, the Tools panel allows you to modify the colors that are used to draw strokes and fills.
- ✓ The Stroke color is applied to lines and shape outlines.
- ✓ The Line, Pen, Oval, Rectangle, PolyStar, and Pencil tools all produce a stroke. The Fill color is applied to the interiors of shapes.
- ✓ The Pen (for closed paths), Oval, Rectangle, PolyStar and Brush tools all produce fills.
- ✓ Practice using the various drawing tools on the stage and note the behavior of each tool that you use.
- ✓ Remember, you can quickly delete everything on the screen by double clicking the Eraser tool.

## 8. Case Study-1

**Problem1: To create an animation to represent the growing moon.**

**Steps to follow:**

1. Open **flash 8** software -> click on **flash document**->go to **windows**->**properties** ->select the **properties** tool-> choose the **Background** to black.
2. Go to **fill color** under tool bar-> select the white color.
3. Select the **oval tool** in order to draw the moon. u will get a white circle.
4. Select the **oval tool** in order to draw the moon. u will get a white circle.
5. Select the white circle on the worksheet using the **selection tool** ->right click ->**converts to symbol**->select **movie clip**->give suitable name e.g.: moon->click **ok**.
6. Go to **filter**->click on the + symbol->select **glow** to apply glowing effect-> select the **color** to white under **glow** and adjust the **blur x/blur y** values.
7. Click on the + symbol again and chose **blur**-> again adjust the **blur x/blur y** values.
8. Place the moon where ever you want on the work area. Double click on **layer1** and rename as **MOON**.
9. **Insert** another layer->renames it as **Animation**.
10. Select the **fill color** to black-> select **oval tool** and draw a circle on the moon to cover the moon->select the newly added circle-> right click-> **convert to symbol**-> **movie clip**-> name it as **Animation**.
11. Go to **filter**-> select + symbol->give the **glow** and **blur** effect as did for moon.
12. Select the 150th frame in moon layer->right click->**insert key frame**. Repeat the same for Animation layer.
13. Click on the 149th key frame of animation layer ->right click->press **create motion**-> select the animation movie clip and move slowly across the moon.
14. Finally go to **control menu** and click on **test movies** you will get a growing moon as the output.

**Problem2: To create an animation to indicate a ball bouncing on steps**

**Steps to follow:**

1. Go to **start- macromedia**- click on **flash document**

2. Select the **line** tool and draw the steps. Colour it using the paint bucket tool
3. Select the **circle** from the tool bar and create a circle on the work area.
4. Now fill the colour to the circle using the **paint bucket tool** from the tool bar.
5. Go to **frames** right click on the first frame and choose **insert key frame**. Slightly move the ball.
6. Repeat the same procedure by adding new key frames to show the ball change the shape of the ball slightly when it touches the surface.
7. In order to change the shape use the **free transform tool**.
8. Go to **control menu** and click on **test movies** .you will observe the ball bouncing on steps.

### **Problem3: To simulate movement of a cloud.**

#### **Steps to follow:**

1. Go to **start- macromedia-** click on **flash document**
2. Create a blue background in **layer1**
3. Now insert a **layer2** and draw the clouds in this layer.
4. In order to create the clouds, go to **tool bar** and select **pencil** option, draw the cloud in **layer2** fill the colour to the cloud, right click on it- choose **convert to symbol** option give the name as cloud
6. Select the **movie clip** option and click **ok**.
7. Go to **filter->**click on the **+** symbol->select **glow** to apply glowing effect-> select the colour to white under **glow** and adjust the **blur x/blur y** values.
9. Give the appropriate **blur** effect to the cloud.
10. Go to **frames**, insert key frame on both the layer, create the motion tween on 2nd layer and move the clouds
11. Finally go to **control menu** and click on **test movies**.

### **Problem4: To draw the fan blades and to give proper animation.**

#### **Steps to follow:**

1. Go to **start-> macromedia->** click on **flash document**
2. Create a background on **layer1**.
3. Insert another layer-> draw only fan blades and its circle.

4. Insert another layer and draw fan stand.
5. On each layer right click on frames and insert **key frames**.
6. Select the fan blade's layer and insert new key frame-> select the fan blades by **free transform** tool and **rotate** the circle a little bit.
7. Repeat the rotation until you get the fan rotation animation
8. Go to **control menu** and click on **test movies** to see the animation.

**Problem5: To display the background given (filename: Tulip.jpg) through your name.**

**Steps to follow:**

1. Go to **start-> macromedia->** click on **flash document**
2. Go to **file-> import->open external library->** select a background image Click **open**.
3. The selected image will be stored in your library. Open library and **drag** the image on the work area by selecting the image.
4. Resize the image to fit on the work area.
5. Select the **text tool** from the **tool bar**.
6. Type your name. Select the text and go to property to apply appropriate font effects like font size, style and colour etc.
7. Go to **control menu** and click on **test movies** to see the final output.

**Problem6: To simulate a ball hitting another ball.**

**Steps to follow:**

1. Go to **start->Macro Media**-click on **Flash document**.
2. Choose the **circle** option displayed in the **toolbar**. Create two circles at the opposite ends.
3. Go to **frames->** right click on the 1st blank frame and click insert **key frames**.
4. Select the 1st ball and make it to move towards the other till it touches.
5. Change the shape of the ball using **free transform tool** as soon as the two balls touches each other. After hitting each other make them to move towards opposite direction.
6. before moving to the opposite direction bring back the balls to its original shapes.
7. Finally test the animation by selecting **control menu ->movie clip**.

## 9. Basic Tools for Adobe Photoshop Professional CS5



### Rectangular Marquee Tool (M):

- ✓ Use this tool to make selections on your image, in a rectangular shape.
- ✓ This changes the area of your image that is affected by other tools or actions to be within the defined shape.
- ✓ Holding the [Shift] key while dragging your selection, restricts the shape to a perfect square.
- ✓ Holding the [Alt] key while dragging sets the center of the rectangle to where your cursor started.



### Move Tool (V)

- ✓ Use this tool to, well, move things.
- ✓ Usually you use it to move a Layer around after it has been placed.
- ✓ Hold the [Shift] key to limit the movements to vertical/horizontal.



### Polygon Lasso Tool (L)

- ✓ Ok, this should be the Lasso Tool, but I use the Polygon Lasso a lot more often.
- ✓ Use this to draw selections in whatever shape you would like.
- ✓ To close the selection, either click on the beginning point (you'll see the cursor change when you're on it), or just double-click.
- ✓ When holding the [Ctrl] key, you'll see the cursor change, and the next time you click, it will close your selection.



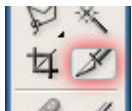
### **Magic Wand Tool (W)**

- ✓ Use this to select a color range.
- ✓ It will select the block of color, or transparency, based on wherever you click.
- ✓ In the Options Bar at the top, you can change the Tolerance to make your selections more/less precise.



### **Crop Tool (C)**

- ✓ The Crop Tool works similarly to the Rectangular Marquee tool (see above if you have no short-term memory).
- ✓ The difference is when you press the [Enter/Return] key; it crops your image to the size of the box.
- ✓ Any information that was on the outside of the box is now gone.
- ✓ Not permanently, you can still undo.



### **Slice Tool (K)**

- ✓ This is used mostly for building websites, or splitting up one image into smaller ones when saving out.
- ✓ It's kind of an advanced tool, and since you're in here for the basics, we'll kind of skip over it.
- ✓ Kind of makes you mad I made you read all that for nothing, huh?



### **Healing Brush Tool (J)**

This is a really useful tool. Mildly advanced

- ✓ You can use this tool to repair scratches and specs and stuff like that on images.
- ✓ It works like the Brush tool (see below).
- ✓ You choose your cursor size, then holding the [Alt] key; you select a nice/clean area of your image.
- ✓ Let go of the [Alt] key and paint over the bad area.
- ✓ It basically copies the info from the first area to the second, in the form of the Brush tool. Only, at the end, it averages the information, so it blends.



### **Brush Tool (B)**

This is one of the first tools ever.

- ✓ It's what Photoshop is based off of.
- ✓ Well, not really, but it's pretty basic.
- ✓ It paints on your image, in whatever color you have selected, and whatever size you have selected.
- ✓ There are a lot of options for it, but this is basic, so you don't get to learn them.



### **Clone Stamp Tool (S)**

- ✓ This is very similar to the Healing Brush Tool (see above).
- ✓ You use it the exact same way, except this tool doesn't blend at the end.
- ✓ It's a direct copy of the information from the first selected area to the second.

- ✓ When you learn to use both of these tools together in perfect harmony, you will be a Photoshop MASTA! Not really, it's just less irritating.



### **History Brush Tool (H)**

- ✓ This tool works just like the Brush Tool (see above) except the information that it paints with is from the original state of your image.
- ✓ If you go Window>History, you can see the History Palette.
- ✓ The History Brush tool paints with the information from whatever History state is selected.



### **Eraser Tool (E)**

This is the anti-Brush tool.

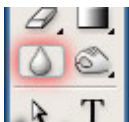
- ✓ It works like an eraser and erases whatever information wherever you click and drag it. If you're on a Layer, it will erase the information transparent.
- ✓ If you are on the background layer, it erases with whatever secondary color you have selected.



### **Gradient Tool (G)**

You can use this to make a gradiation of colors.

- ✓ Gradiation doesn't appear to be a word, but it makes sense anyway.
- ✓ It creates a blending of your foreground color and background color when you click and drag it. Like a gradient.





### **Blur Tool (R)**

- ✓ The Blur tool is cool.
- ✓ It makes things blurry.
- ✓ Click and drag to make things blurry.
- ✓ The more you click and drag, the blurrier things get.



### **Dodge Tool (O)**

- ✓ This tool isn't as crappy as the car brand.
- ✓ It's actually used to lighten whatever area you use it on.
- ✓ As long as it is not absolute black. Absolute black won't lighten.



### **Path Selection Tool (A)**

- ✓ You use this tool when working with paths.
- ✓ Since this is all about the basics, I won't go into details.
- ✓ It's related to the Pen Tool (see below) though.



### **Horizontal Type Tool (T)**

- ✓ It makes type. Or text. Or whatever you want to call it.
- ✓ You can click a single point, and start typing right away.
- ✓ Or you can click and drag to make a bounding box of where your text/type goes.
- ✓ There are a lot of options for the Type Tool. Just play around, it's fairly straight-forward.



### **Pen Tool (P)**

- ✓ I mentioned this tool above. It's for creating paths, in which you would use the Path Selection Tool to select the path.
- ✓ Paths can be used in a few different ways, mostly to create clipping paths, or to create selections.
- ✓ You use the tool by clicking to add a point.
- ✓ If you click and drag, it will change the shape of your path, allowing you to bend and shape the path for accurate selections and such.



### **Rectangle Tool (U)**

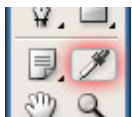
- ✓ By default it draws a Shape Layer in the form of a rectangle.
- ✓ It fills the rectangle with whatever foreground color you have selected.
- ✓ It's pretty complicated; don't hurt yourself with this one.



### **Notes Tool (N)**

Like post-it notes, but digital.

- ✓ You can use this tool to add small little note boxes to your image.
- ✓ These are useful if you're very forgetful or if you're sharing your Photoshop file with someone else. I'm pretty sure it only works with .PSD files.



### **Eyedropper Tool (I)**

- ✓ This tool works by changing your foreground color to whatever color you click on.

- ✓ Holding the [Alt] key will change your background color.



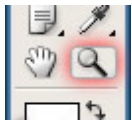
### **Hand Tool (T)**

You can really make short work of your job with the Hand Tool.

It's for moving your entire image within a window.

So if you're zoomed in and your image area is larger than the window, you can use the Hand Tool to navigate around your image.

Just click and drag. You can get to this tool at any time when using any other tool by pressing and holding the [Spacebar].



### **Zoom Tool (Z)**

Pretty obvious what this tool does.

- It allows you to zoom into your image. Don't be dumb, it doesn't actually change the size of your image.
- Hold the [Alt] key to zoom out. Holding the [Shift] key will zoom all of the windows you have open at the same time.
- Double-click on the Zoom Tool in the palette to go back to 100% view.

## **10.Case Study-2**

### **Problem1: Image editing using adobe Photoshop (filter)**

To develop a filter image using adobe Photoshop

#### **Steps to follow:**

1. Open Adobe Photoshop CS5.
2. Select the file menu -> click open -> browse a picture
3. Select filter menu from the title bar
4. Select filter option -> blur -> radial blur

5. Set the appropriate radial to blur the picture and click ok
6. Finally save the image file.

**Problem2: Image editing using adobe Photoshop (pattern)**

To generate a repeated pattern of image using adobe Photoshop

**Steps to follow:**

1. Open Adobe Photoshop CS5.
2. Select the file menu and then click open for a sample picture
3. Select filter menu from the menu bar
4. Choose pattern maker from the filter menu
5. New window will be opened and the browsed image will be present
6. Select a pattern of the picture and click generate
7. Finally a pattern of image will be displayed.
8. Finally save the image file.

**Problem3: to prepare a cover page for the book in your Subject area. Plan your own design.**

1. Open **Adobe Photoshop CS5**-> **file**-> **new**-> enter **height 500** and **width 400** for the cover page.
2. Select the **rectangle tool** in the **tool bar** and draw on the half of the work area-> colour it
3. Repeat the same for remaining area-> use different colours to colour.
4. Copy any picture of ur choice and place it on the work area->resize it using **free transform tool**.
5. Select the **text tool** and type text of your choice.
6. Apply the text font size, colour and style of your choice.
7. Go to **layer**->**layer style**->**blended option**-> select **glow** options of your choice.
8. Apply the effects using **blended options**.
9. Finally save the image file.

**Problem4: To design a visiting card containing at least one Graphic and text information.**

**Steps to follow:**

1. Open **Adobe Photoshop CS5**-> **file**-> **new**-> enter **height 200 and width 400** for the visiting card.
2. Select the **rectangle tool** in the **tool bar** and **draw** on the half of the work area-> colour it.  
Repeat the same for remaining half-> use different colours to colour.
3. Copy any picture of your choice and place it on the work area-> Resize it using **transforms tool**.
4. Select the **text tool** and type text of your choice.
5. Apply the text **font size, colour and style** of your choice.
6. Finally test the animation by selecting **control menu and test movies** to see your visiting cards.

**Problem5: To take a photographic image. Give a title for the Image. Put the border. Write your names. Write the name of Institution and place.**

**Steps to follow:**

1. Open **Adobe Photoshop CS5**-> **file**-> **new**-> enter **height 800 and width 600**
2. Open an image file and copy the image->**paste** the copied image on the new file.
3. Right click on the **rectangle tool**->**custom shape**-> select the **shape**->select the colour- >**drag** on your image.
4. Select the **text tool**->type your **name, institution name and place**.
5. Finally save the image file.

**Problem6: To extract the flower only from given Photographic image and organize it on a background. Selecting your own background for organization**

**Steps to follow:**

1. Open **Adobe Photoshop CS5**-> **file**->**open**-> choose a file and open it.
2. Select the flower from the image using the **lasso tool**.
3. Go to **edit**-> **copy**->Again go to **file**->**new**->give **height 500 and width 500**.
4. Choose appropriate background and foreground colour from the tool bar.
5. Go to **edit**->**fill**->under **use** select **background colour**->**ok**.
6. Go to **edit**->**paste**.

7. Finally save the image file.

**Problem7: to adjust the brightness and contrast of the picture so that it gives an elegant look.**

**Steps to follow:**

1. Open **Adobe Photoshop CS5**-> **file->open**-> choose a file and open it.
2. Go to **image->adjustments->Brightness/Contrast**.
3. After getting the Brightness/Contrast window adjust the brightness and contrast by dragging the appropriate bar setting.
4. Finally save the image file.

**Problem8: to position the picture preferably on a plain background of a colour of your choice - positioning includes rotation and scaling.**

**Steps to follow:**

1. Open **Adobe Photoshop 7.0**-> **file->open**-> choose a file and open it.
2. Select the flower from the image using the **lasso tool**.
3. Go to **edit->copy**->Again go to **file->new**->give **height 500** and **width 500**.
4. Choose appropriate background and foreground colour from the tool bar.
5. Go to **edit->fill**->under **use** select **background colour**->**ok**.
6. Go to **edit->paste**->again go to **edit->free transform tool**-> you will get a box around the image for scaling and rotating.
7. **Rotate** and **scale** as per your requirement->and press **apply**.
8. Save the image.

## **Computer Graphics**

### **1. How do we use Borland Graphics Interface (graphics.h)?**

For those of you migrating from Borland, you may be wondering where graphics.h is. Unfortunately, graphics.h is a Borland specific library and cannot be used with Dev-C++. Fortunately, a benevolent soul by the name of Michael Main has modified a BGI emulation library for Windows applications to be used under MinGW (and therefore Dev-C++) which he has aptly named WinBGIm. The files we need are:

**graphics.h (download to C:\Dev-Cpp\include)**

**libbgi.a (download to C:\Dev-Cpp\lib)**

After you have downloaded the files to the correct locations, you can now use WinBGIm's graphic.h as you would Borland's graphics.h with a few caveats.

#### ***Using library files:***

**Step1: You have to tell Dev-C++ where to find the library functions that WinBGIm references this is done in the "Project Options" dialog box.**

Here are instructions on how to do this with a new project: Follow step 2 and step 3 of "Using Dev-C++".

#### **Step 2: Create a new project.**

A "project" can be considered as a container that is used to store all the elements that are required to compile a program.

- ✓ Go to the "File" menu and select "New", "Project..."
- ✓ Choose "Empty Project" and make sure "C++ project" is selected. Here you will also give your project a name. You can give your project any valid filename, but keep in mind that the name of your project will also be the name of your final executable.
- ✓ Once you have entered a name for your project, click "OK".
- ✓ Dev-C++ will now ask you where to save your project.

#### **Step 3: Create/add source file(s).**

You can add empty source files one of two ways:

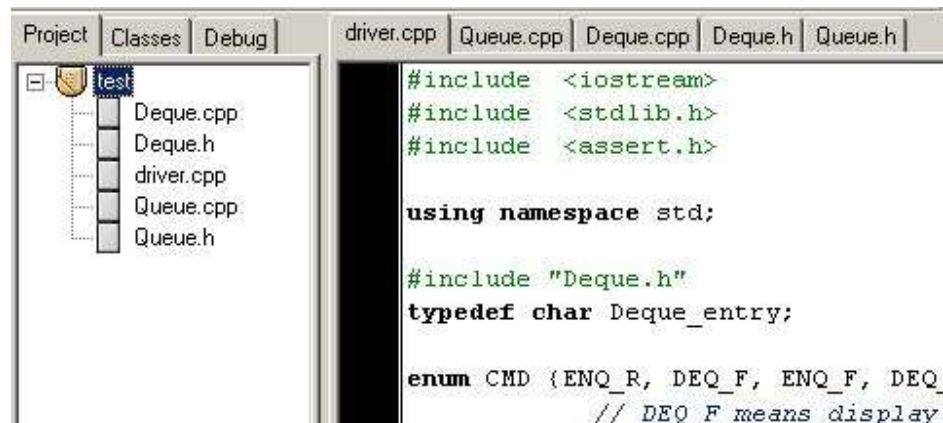
- ✓ Go to the "File" menu and select "New Source File" (or just press CTRL+N) OR
  - ✓ Go to the "Project" menu and select "New File".
- Note that Dev-C++ will not ask for a filename for any new source file until you attempt to:

1. Compile
2. Save the project
3. Save the source file
4. Exit Dev-C++

You can add pre-existing source files one of two ways:

- ✓ Go to the "Project" menu and select "Add to Project" OR
- ✓ Right-click on the project name in the left-hand panel and select "Add to Project".

EXAMPLE: Multiple source files



In this example, more than 3 files are required to compile the program; The "driver.cpp" file references "Deque.h" (which requires "Deque.cpp") and "Deque.cpp" references "Queue.h" (which requires "Queue.cpp").

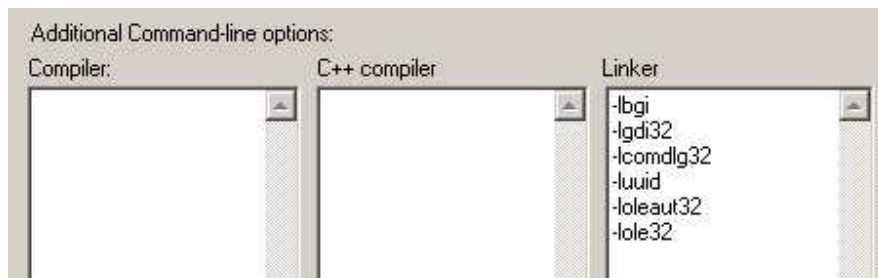
- ✓ Go to "Project" menu and choose "Project Options" (or just press ALT+P).
- ✓ Go to the "Parameters" tab



- ✓ In the "Linker" field, enter the following text:

**-lbgi**  
**-lgdi32**  
**-lcomdlg32**  
**-luuid**  
**-loleaut32**  
**-ole32**

Project Options -> Parameters:



- ✓ Click "OK".
- ✓ Follow step 4, step 5 and step 6 of "Using Dev-C++".

#### Step 4: Compile.

Once you have entered all of your source code, you are ready to compile.

- Go to the "Execute" menu and select "Compile" (or just press CTRL+F9).

It is likely that you will get some kind of compiler or linker error the first time you attempt to compile a project. Syntax errors will be displayed in the "Compiler" tab at the bottom of the screen. You can double-click on any error to take you to the place in the source code where it occurred. The "Linker" tab will flash if there are any linker errors. Linker errors are generally the result of syntax errors not allowing one of the files to compile.

Once your project successfully compiles, the **"Compile Progress"** dialog box will have a status of **"Done"**. At this point, you may click **"Close"**.

#### Step 5: Execute.

You can now run your program.

- Go to the "Execute" menu, choose "Run".

Note: to pass command-line parameters to your program, go to the **"Execute"** menu, choose **"Parameters"** and type in any parameters' you wish to pass.

### **Step 6: Debug.**

When things aren't happening the way you planned, a source-level debugger can be a great tool in determining what really is going on. Dev-C++'s basic debugger functions are controlled via the "Debug" tab at the bottom of the screen; more advanced functions are available in the "Debug" menu.

#### **Using the debugger:**

The various features of the debugger are pretty obvious. Click the **"Run to cursor"** icon to run your program and pause at the current source code cursor location; Click **"Next Step"** to step through the code; Click **"Add Watch"** to monitor variables.

Setting breakpoints is as easy as clicking in the black space next to the line in the source code. See the Dev-C++ help topic "Debugging Your Program" for more information.

#### **BGI and WinBGIm differences:**

WinBGIm is a superset of BGI and as such may have functions and features with which you are unfamiliar. See Michael Main's BGI Documentation for more information.

### **2. Test program:**

Just to make sure you've got everything set up correctly, try this test code in a new Dev-C++ WinBGIm project:

```
#include <graphics.h>

int main()
{
    initwindow(400,300); //open a 400x300 graphics window

    moveto(0,0);

    lineto(50,50);

    while(!kbhit()); //wait for user to press a key

    closegraph(); //close graphics window

    return 0; }
```

### 3. Algorithms

#### 3.1 Line Drawing Algorithms

**Aim:**

- ✓ To study and Implement DDA Line Drawing Algorithm
- ✓ To study and Implement Bresenham 's Line Drawing Algorithm
- ✓ To study and Implement Midpoint Line Drawing Algorithm.

**Pre-requisite:** Knowledge of

- ✓ Point Plotting Methods
- ✓ Graphics Initializations and
- ✓ DDA Algorithm, Bresenham's Algorithm and Midpoint Algorithms

##### 3.1.1 Steps for Line Drawing using DDA algorithm when $|m| < 1$ :

Step1: Load the line end points are  $(x_1, y_1)$  and  $(x_2, y_2)$ .

Step2: Calculate  $\Delta x = |x_2 - x_1|$  and  $\Delta y = |y_2 - y_1|$

Step3: Determine length

If  $\text{abs}(\Delta x) \geq \text{abs}(\Delta y)$  then

Length =  $\text{abs}(\Delta x)$  else

Length =  $\text{abs}(\Delta y)$

Step4: Select the increment of  $\Delta x$  (or)  $\Delta y$  to be one raster unit

New\_  $\Delta x = (x_2 - x_1) / \text{Length}$  and

New\_  $\Delta y = (y_2 - y_1) / \text{Length}$

**Note:** Round the values rather than truncate, so that center pixel addressing is handled correctly.

/\* Initial raster unit \*/

Sign returns -1, 0, 1 for arguments  $<0, =0, >0$  respectively approximate the line length.

$X = x1 + .5 * \text{sign}(\text{new\_}\Delta x)$

$Y = y1 + .5 * \text{sign}(\text{new\_}\Delta y)$

Step5: loop will start

$i = 1$

While ( $i \leq \text{Length}$ )

setpixel(int(X), int(Y))

$X = x + \text{new\_}\Delta x$

$Y = y + \text{new\_}\Delta y$

$i = i + 1$

Repeat loop upto  $\leq \text{Length}$  times

Step6: stop.

### **3.1.2 Steps for Bresenham's Line Drawing Algorithm when $|m| < 1$ :**

1. Input the two line endpoints and store the left endpoint in ( $x1, y1$ )
2. Load ( $x1, y1$ ) into the frame buffer; that is, plot the first point.
3. Calculate constants  $\Delta x$ ,  $\Delta y$ ,  $2\Delta y$  and  $2\Delta y - 2\Delta x$ , and obtain the starting value for the decision parameter as:

$$P_0 = 2\Delta y - \Delta x$$

4. At each  $X_k$ , the next point the line, starting at  $k=0$ , perform the following test:
  - a). If  $P_k < 0$ , the next point to plot is ( $X_k + 1, Y_k$ ) and  $P_{k+1} = P_k + 2\Delta y$
  - b). Otherwise, the next point to plot is ( $X_k + 1, Y_k + 1$ ) and  $P_{k+1} = P_k + 2\Delta y - 2\Delta x$
5. Repeat step4  $\Delta x$  times.

### **3.1.3 Steps for Midpoint Line-Drawing Algorithm when $|m| < 1$ :**

Step1: Input the two line endpoints and store the left endpoint in ( $x1, y1$ )

Step2: Load ( $x1, y1$ ) into the frame buffer; that is, plot the first point.

Step3: Calculate constants  $\Delta x = (x_2 - x_1)$ ,  $\Delta y = (y_2 - y_1)$  and obtain the starting value for the decision parameter as; ( $\Delta x = w$ ;  $\Delta y = h$ )

$F_0 = h(ax + 1 - ax) - w(ay + \frac{1}{2} - ay) = \Delta y - \Delta x / 2$  This is obtained from when the  $F(x,y)=0$

Step4: At each  $X_k$  along the line, starting at  $k=0$ , Perform the following test;

If  $F_k < 0$ , Let assume  $Y_{k+1} = Y_k$ ; the next point to plot is  $(X_k + 1, Y_k)$  Then  $F_{k+1} = F_k + \Delta y$   
Otherwise, If  $F_k \geq 0$ , Let assume  $Y_{k+1} = Y_k + 1$  then  $F_{k+1} = F_k + \Delta y - \Delta x$

Step5: Repeat step4  $\Delta x$  times.

Step6: Stop

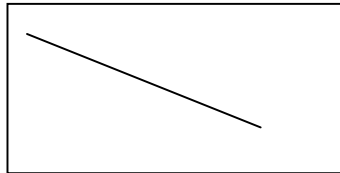
**Sample Input:**

Enter the two end points of the line:

Enter an Initial Point ( $x_1, y_1$ ): - 100 200

Enter the Final Point( $x_2, y_2$ ): - 200 300

**Sample Output:**



### 3.2 Circle Drawing Algorithms

**Aim:**

- ✓ To study and implement DDA circle algorithm given the points of the centre and the radius.
- ✓ To study and implement Bresenham's circle generating algorithm given the points of the centre and the radius.
- ✓ To study and implement Midpoint circle generating algorithm given the points of the centre and the radius.

**Pre-requisite:** Knowledge of

- ✓ Basic Representation of 8 way symmetric method for circle generation

- ✓ Basic Representation trigonometric and polynomial methods
- ✓ Midpoint circle algorithm

### 3.2.1 Steps for DDA Circle drawing algorithm:

1. Read the radius( $r$ ) of the circle and calculate value of epsilon

2. Initialize starting point

Start\_x=0

Start\_y=r

3.  $x_1 = \text{Start\_x}$

$y_1 = \text{Start\_y}$

4. do

{

$x_2 = x_1 + \text{epsilon}(y_1)$

$y_2 = y_1 - \text{epsilon}(x_2)$

Where  $x_2$  represents  $x_{n+1}$  and  $x_1$  represents  $x_n$

Plot (int( $x_2$ ), int( $y_2$ ))

$x_1 = x_2;$

$y_1 = y_2;$

[Reinitialize the current point]

} while ( $y_1 - \text{start\_y} < \text{epsilon}$  or  $(\text{start\_x} - x_1) > \text{epsilon}$ )

**Note:** Check if the current point is the starting point or not. If the current point is not the starting point repeat step4; otherwise go to step 5

5. Stop.

Determine remaining part of circle can be drawn by reflecting point about y axis, x axis and about origin.

### 3.2.2 Steps for Bresenham's Circle drawing algorithm:

1. Read the radius( $r$ ) of the circle

2. Initialize starting point

$x=0;$        $y=r$

3. Initial decision variable as;

$d_i=3-2r$

4. do

{

$\text{plot}(x,y)$

    if( $d_i < 0$ ) then

        {  $d_{i+1}=d_i+4x+6$

    }

    else

        {  $d_{i+1}=d_i+4(x-y)+10$

$y=y-1$

    }

$x=x+1$

} while( $x < y$ )

5. Determine remaining part of circle can be drawn by reflecting point about y axis, x axis and about origin.

$\text{Plot}(y, x); \text{Plot}(y, -x); \text{Plot}(x, -y); \text{Plot}(-x, -y); \text{Plot}(-y, -x); \text{Plot}(-y, x); \text{Plot}(-x, y)$

6. Stop

### 3.2.3 Steps for Midpoint Circle drawing algorithm:

1. Read the radius( $r$ ) of the circle

2. Initialize starting point

$x=0; y=r$

3. Initial decision variable

$d_i=1.25-r$

4. do

{

plot(x,y)

if( $d_i < 0$ ) then

{

$x=x+1$

$y=y-1$

$d_{i+1}=d_i+2x+2$

}

else

{

$x=x+1$

$y=y-1$

$d_{i+1}=d_i+2x+2y+1$

}

} while( $x < y$ )

5. Determine remaining part of circle can be drawn by reflecting point about y axis, x axis and about origin.

Plot(y, x); Plot(y,-x); Plot(x,-y); Plot (-x,-y); Plot (-y,-x); Plot (-y, x); and Plot (-x, y)

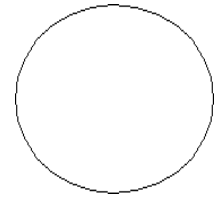
6. Stop



**Sample Input and Output:**

Enter the coordinates of the centre: - x-coordinate = 350 & y-coordinate = 250

Enter the radius: - 50

**3.3 Polygon Filling Algorithms**

**Aim:** To study and Implement Polygon Filling Algorithms

Pre-requisite: Knowledge of Polygon Filling Algorithms

- ✓ Boundary fill Algorithm(4 connected and 8 connected)
- ✓ Scan line polygon filling Algorithm.
- ✓ Flood Fill Algorithm (x, y) are the interior points, boundary is the boundary color and fill color is the color to be filled.

**3.3.1 Steps for Basic Filling Algorithm**

1. Set the user specified point.
2. Store the four neighboring pixels in a stack.
3. Remove a pixel from the stack.
4. If the pixel is not set,
  - Set the pixel
  - Push its four neighboring pixels into the stack
5. Go to step 3
6. Repeat till the stack is empty.

Basic Filling Algorithm (Code)

```
void fill(int x, int y)
{ if(getPixel(x,y)==0)
{ setPixel(x,y);
  fill(x+1,y);
```

```
fill(x-1,y);  
fill(x,y+1);  
fill(x,y-1);}}
```

### **3.3.2 Steps for Boundary Fill Algorithm:**

1. For filling a region with a single boundary color.
2. Condition for setting pixels:

Color is not the same as border color

Color is not the same as fill color

### **3.3.3 Boundary Fill Algorithm Code:**

```
Void boundary Fill (int x, int y, int fillColor, int borderColor)  
{  
    getPixel(x, y, color);  
    if ((color!= borderColor) && (color!= fillColor))  
    { setPixel(x,y);  
      boundaryFill(x+1,y,fillColor,borderColor);  
      boundaryFill(x-1,y,fillColor,borderColor);  
      boundaryFill(x,y+1,fillColor,borderColor);  
      boundaryFill(x,y-1,fillColor,borderColor);  
    }  
}
```

### **3.3.4 Steps for Flood Fill Algorithm:**

1. For filling a region with multiple boundary colors.
2. Condition for setting pixels

Color is same as the old interior color

```

void floodFill(int x, int y, int fillColor, int oldColor)
{
    getPixel(x, y, color);
    if (color == oldColor)
    {
        setPixel(x,y);
        floodFill(x+1, y, fillColor, oldColor);
        floodFill(x-1, y, fillColor, oldColor);
        floodFill(x, y+1, fillColor, oldColor);
        floodFill(x, y-1, fillColor, oldColor);
    }
}

```

### Sample Output

Enter your choice

1. Boundary Fill
2. Scan line polygon filling algorithm.
3. Exit

### 3.4 Basic Transformation methods

**Aim:** To implement set of basic transformations on polygon i.e. Translation, Rotation, Scaling, Reflection and Shearing.

**Pre-requisite:** Knowledge of matrix fundamentals and basic transformations on polygon i.e. translation, rotation, scaling, reflection and shearing.

**Description** : Transformations allows us to uniformly alter the entire picture. The geometric transformations considered here- translation, scaling and rotation are expressed in terms of matrix multiplication. Homogeneous coordinates are considered to uniformly treat the translations.

**3.4.1 Scaling Transformation:** A 2D point can be scaled by multiplication of the coordinate values (x,y) by scaling factors **Sx** and **Sy** to produce the transformed coordinates (x',y').

Matrix format:  $[X'] = [X][S]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[S]$  - Scaling Matrix

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}$$

**3.4.2 Translation Transformation:** A 2D point can be translated by adding the coordinate values (x,y) by Translation distances tx and ty to produce the transformed coordinates (x',y').

Matrix format:  $[X'] = [X][T]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[T]$  - Translation Matrix

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} tx & ty \end{bmatrix}$$

**3.4.3 Rotation Transformation:** A 2D point can be rotated by repositioning it along a circular path in the xy plane. We specify the rotation angle and the position of the rotation point about which the object is to be rotated. Multiplication of the coordinate values (x,y) by rotation matrix produce the transformed coordinates (x',y').

Matrix format:  $[X'] = [X][R]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[R]$  - Rotation Matrix

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

**3.4.4 Reflection Transformation:** Reflection is a transformation that produces a mirror image of an object.

a). Transformation matrix for reflection about the line y=0, is

Matrix format:  $[X'] = [X][\text{Refy}]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[\text{Refy}]$  – Reflection Matrix

$$[x' y'] = [x y] \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b). Transformation matrix for reflection about the line x=0, is

Matrix format:  $[X'] = [X][\text{Refx}]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[\text{Refx}]$  – Reflection Matrix

$$[x' y'] = [x y] \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**3.4.5 Shearing Transformation:** Shearing is a transformation that distorts the shape of an object.

An X-direction shear relative to the x axis is produced by the transformation matrix is:

Matrix format:  $[X'] = [X][\text{Shy}]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[\text{Shy}]$  – Shearing Matrix

$$[x' y'] = [x y] \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A y-direction shear relative to other reference lines is produced by the transformation matrix is:

Matrix format:  $[X'] = [X][\text{Shy}]$

Where  $[X']$  - Transformed Matrix;  $[X]$  - Point or Object Matrix; and  $[\text{Shy}]$  – Shearing Matrix

$$[x' y'] = [x y] \begin{bmatrix} 1 & shx & -shx y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Sample Input and Output:

Enter the no. of edges:-4

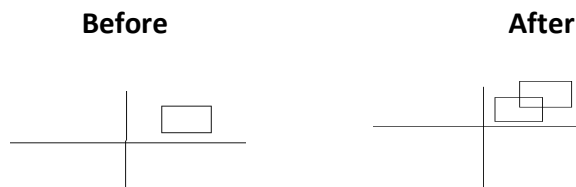
Enter the co-ordinates of vertex 1:- 30, 30

Enter the co-ordinates of vertex 2:- 30, 90

Enter the co-ordinates of vertex 3:- 90, 90

Enter the co-ordinates of vertex 4:- 90, 30

Enter the Translation factor for x and y:-20, 20



### 3.5 Rotation about an arbitrary point

**Aim:** To study and Implement of rotation about an arbitrary point.

Pre-requisite; Knowledge of rotation about an arbitrary point (xp,yp).

**3.5.1 Rotation about an arbitrary point (xp, yp):** This is done by three transformation steps: translation of the arbitrary point (xp,yp) to the origin, rotate about the origin, and then translate the center of rotation back to where it belongs. To transform a point, we would multiply all the transformation matrices together to form an overall transformation matrix.

1. The translation which moves (xp,yp) to the origin:

$$T1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix}$$

2. The rotation about the origin:

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

3. The translation to move the center point back to the original position

$$T2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

Matrix format:  $[X'] = [X][T1][R][T2]$

Where

[X'] - Transformed Matrix;

[X1] – Object or Point Matrix;

[T1] – Translation1 Matrix;

[T2] – Translation2 Matrix

[R] – Rotation Matrix

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xc & -yc & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xc & yc & 1 \end{bmatrix}$$

Sample Input and Output 1:

Enter the no. of edges:-4

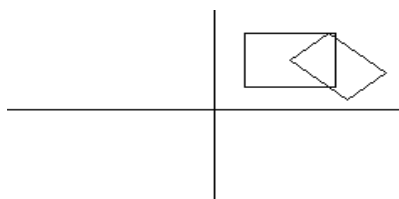
Enter the x and y co-ordinates:- 30, 30

Enter the x and y co-ordinates:- 30, 90

Enter the x and y co-ordinates:- 90, 90

Enter the x and y co-ordinates:- 90, 30

Enter the Rotation angle:45 Degree



### 3.6 Line Clipping Algorithm

**Aim:** To study and implement Line Clipping Algorithm using Cohen Sutherland & Subdivision and to study and implement Line Clipping Algorithm using Liang Barsky

**Pre-requisite:** Knowledge of Nine region code, Line Clipping Algorithms using Cohen Sutherland & Subdivision, and Liang Barsky Algorithm

### 3.6.1 Steps for Cohen-Sutherland Line Clipping Algorithm;

1. Read two corners of the window say ( $Wx1, Wy1$ ) and ( $Wx2, Wy2$ )
2. Read two end points of the line say P1 ( $x1, y1$ ) and P2 ( $x2, y2$ )
3. Assign the region codes for two end points P1 and P2 using following steps

Initialize code with bits 0000

Set Bit1 → if ( $X < Wx1$ )

Set Bit2 → if ( $X > Wx2$ )

Set Bit3 → if ( $Y < Wy1$ )

Set Bit4 → if ( $Y > Wy2$ )

4. Check for visibility of line P1 P2

(a) If region codes for both endpoints P1 and P2 are zero then the line completely visible. Hence draw the line and goto step 8.

(b) If region codes for endpoints are non-zero and the logical ANDing of them is also non-zero then the line completely invisible. So reject the line and goto step 8.

(c) If region codes for two endpoints do not satisfy the condition in 4(a) and 4(b) the line is partially visible. Hence find the intersection value of end points then goto step 5.

5. Determine the intersecting edge of the clipping window by inspecting the region codes of two endpoints.

(a) If region codes for both endpoints are non-zero then find intersection points P1' and P2' with window boundary edges of clipping window with respect to point P1 and point P2 respectively. goto step6

(b) If region codes for any one end point is non-zero then find intersection points P1' (or) P2' with window boundary edges of clipping window with respect to it. goto step6

6. Divide the Line segments considering intersection points
7. Reject the line segment if any one end point of it appears outside the clipping window.
8. Draw the remaining line segments.



9. Stop.

### **3.6.2 Steps for Cohen-Sutherland Subdivision (or) Midpoint Line Clipping Algorithm**

1. Read two corners of the window say ( $Wx1$ ,  $Wy1$ ) for left corner and ( $Wx2$ ,  $Wy2$ ) for right corner.

2. Read two end points of the line say  $P1 (x1, y1)$  and  $P2 (x2, y2)$

3. Assign the region codes for two end points  $P1$  and  $P2$  using following steps

Initialize code with bits 0000

Set Bit1 → if ( $X < Wx1$ )

Set Bit2 → if ( $X > Wx2$ )

Set Bit3 → if ( $Y < Wy1$ )

Set Bit4 → if ( $Y > Wy2$ )

4. Check for visibility of line  $P1 P2$

(a) If region codes for both endpoints  $P1$  and  $P2$  are zero then the line completely visible. Hence draw the line and goto step 6.

(b) If region codes for endpoints are non-zero and the logical ANDing of them is also non-zero then the line completely invisible. So reject the line and goto step 6.

(c) If region codes for two endpoints do not satisfy the condition in 4(a) and 4(b) the line is partially visible. Hence find the intersection value of end points then goto step 5.

5. Divide the partially visible line segments in equal parts and repeats steps 3 through 5 for both subdivided line segments until you get completely visible and completely invisible line segments.

6. Stop

### **3.6.3 Steps for Liang Barsky Line Clipping Algorithm:**

1. Read two corners of the window say ( $Wx1$ ,  $Wy1$ ) for left corner and ( $Wx2$ ,  $Wy2$ ) for right corner.

2. Read two end points of the line say  $P1 (x1, y1)$  and  $P2 (x2, y2)$

3. Calculate the values of the parameter  $P_i$  and  $Q_j$  for  $i = 1, 2, 3, 4$  such that

$$P_1 = -\Delta x; \quad Q_1 = x_1 - Wx_1$$

$$P_2 = \Delta x; \quad Q_2 = Wx_2 - x_1$$

$$P_3 = -\Delta y; \quad Q_3 = y_1 - Wy_1$$

$$P_4 = \Delta y; \quad Q_4 = Wy_2 - y_1$$

3. If  $P_i = 0$  then

{The line is parallel to  $i$ th boundary

Now, if  $Q_i < 0$  then

Line is completely outside the boundary; hence discard the line segment and goto step9

} else

Otherwise,

{

Check whether the line is horizontal or vertical and accordingly check the line endpoint with corresponding boundaries. If line endpoints within the bounded area then use them to draw line otherwise use boundary coordinates to draw line. goto step9

}

4. Initialize values for  $t_1$  and  $t_2$  as

$$t_1 = 0 \text{ and } t_2 = 1$$

5. Calculate values for  $Q_i / P_i$  for  $i = 1, 2, 3, 4$
6. Select values of  $Q_i / P_i$  where  $P_i < 0$  and assign maximum out of them as  $t_1$
7. Select values of  $Q_i / P_i$  where  $P_i > 0$  and assign minimum out of them as  $t_2$

If ( $t_1 < t_2$ )

Calculate the endpoints of the clipped lines as follows:-

$$xx_1 = x_1 + t_1 \Delta x; \quad xx_2 = x_1 + t_2 \Delta x; \quad yy_1 = y_1 + t_1 \Delta y; \quad yy_2 = y_1 + t_2 \Delta y$$

9. Draw line (xx 1, yy1, xx2, yy2)

9. Stop

**Sample Input and Output :**

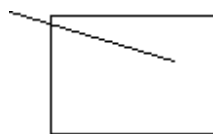
Enter Minimum window co-ordinates:- 200 250

Enter Maximum window co-ordinates:- 300 350

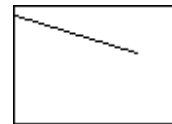
Enter co-ordinates of first point of line: - 180 250

Enter co-ordinates of second point of line: - 200 300

**Before Clipping**



**After Clipping**



### 3.7 Polygon Clipping Algorithm

**Aim:** To study and implement Polygon Clipping Algorithm using Sutherland Hodgeman Algorithm

**Pre-requisite:** Knowledge of Polygon Clipping Algorithm using Sutherland Hodgeman

#### 3.7.1 Steps for Sutherland-Hodgeman polygon clipping Algorithm

1. Read two corners of the window say (Wx1, Wy1) and (Wx2, Wy2)
2. Read edges of the polygon like triangle say P1 (x1, y1), P2 (x2, y2) and P3(x3, y3)
3. Consider the left edge of the window
4. Compare the vertices of each edge of the polygon, individually according to four possible relationships between the edge and the clipping boundary discussed earlier.
5. Repeat the steps 4 and 5 for remaining edges of the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.
6. Stop.

**Sample Input and Output :**

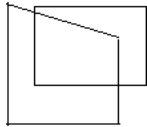
Enter Minimum window co-ordinates:- 200 250

Enter Maximum window co-ordinates:- 300 350

Enter co-ordinates of first point of line: - 180 250

Enter co-ordinates of second point of line: - 200 300

**Before Polygon Clipping**



**After Polygon Clipping**



### 3.8 Curves Generation

**Aim:** 1.To study and implement Curves Generation using Bezeir Curves and B-Splines

**Pre-requisite:** Knowledge of Curves Generation Using Bezeir Curves and B-Splines

#### 3.8.1 Steps for Bezeir Curves algorithm :

1. Get Four control points say A( $x_A$  ,  $y_A$  ), B( $x_B$  ,  $y_B$ ), C( $x_C$  ,  $y_C$ ) ,D( $x_D$  ,  $y_D$  )
2. Divide the curves represented by point A, B, C and D in two sections

$$x_{AB} = (x_A + x_B) / 2$$

$$y_{AB} = (y_A + y_B) / 2$$

$$x_{BC} = (x_B + x_C) / 2$$

$$y_{BC} = (y_B + y_C) / 2$$

$$x_{CD} = (x_C + x_D) / 2$$

$$y_{CD} = (y_C + y_D) / 2$$

$$x_{ABC} = (x_{AB} + x_{BC}) / 2$$

$$y_{ABC} = (y_{AB} + y_{BC}) / 2$$

$$x_{BCD} = (x_{BC} + x_{CD}) / 2$$

$$y_{BCD} = (y_{BC} + y_{CD}) / 2$$

$$x_{ABCD} = (x_{ABC} + x_{BCD}) / 2$$

$$y_{ABCD} = (y_{ABC} + y_{BCD}) / 2$$

3. Repeat the step 2 for section AAB, ABC, and ABCD and section ABCD, BCD, CD and D
4. Repeat step 3 until we have sections so short that they can be replaced by straight lines
5. Replace small sections by straight lines
6. Stop

### 3.8.2 Steps for B-Splines Curves algorithm :

1. The B-spline basis functions are defined recursively as follows:

$$N_{i,1}(u) = 1$$

$$\text{if } t_i \leq u < t_{i+1}$$

$$= 0$$

Otherwise

2. The knot values are chosen with the following rule:

$$t_i = 0$$

$$\text{if } i < k = i - k + 1$$

$$\text{if } k \leq i \leq n$$

$$= n - k + 2 \text{ if } i > n$$

### Sample Input and Output:

Enter the no. of control points: 4

Enter the control point1:- 20 50

Enter the control point2:- 30 10

Enter the control point3:- 40 50

Enter the control point4:- 50 10



## ***References:***

1. Donald Hearn and M.Pauline Baker,"Computer Graphics with C version (or) OpenGL", Second Edition Pearson Education.
2. Amarendra N.Sinha, and Arun D. Udai,"Computer Graphics", TATA McGraw-Hill Publication Limited, New Delhi
3. J.D. Foley, A.V dam, S.K. Feiner, "Computer Graphics: Principles and Practice in C (or) OpenGL", Pearson Education, Second Edition, New Delhi.
4. ISRD Group Lucknow, "Computer Graphics", Tata McGraw-Hill Publication Limited, New Delhi.
5. Rogers and Adams,"Mathematical Elements for Computer Graphics", TMH
6. Xiang and Plastok,"Schaum's Outlines Computer Graphics ", Second Edition, TMH
7. Harrington, "Computer Graphics ", Tata McGraw Hill
8. [http://help.adobe.com/en\\_US/legalnotices/index.html](http://help.adobe.com/en_US/legalnotices/index.html) for flash
9. [www.enthoesweb.com](http://www.enthoesweb.com) for Multimedia Resources