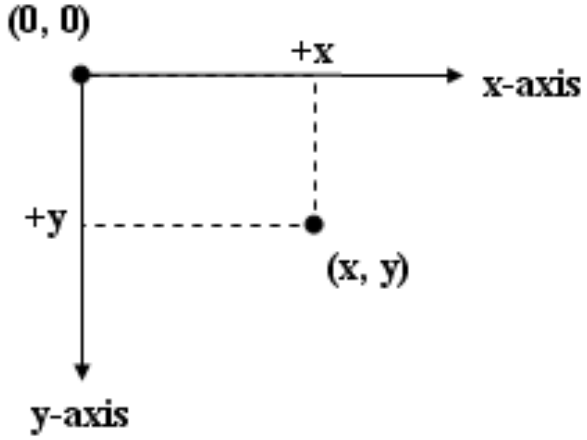
Chapter 6

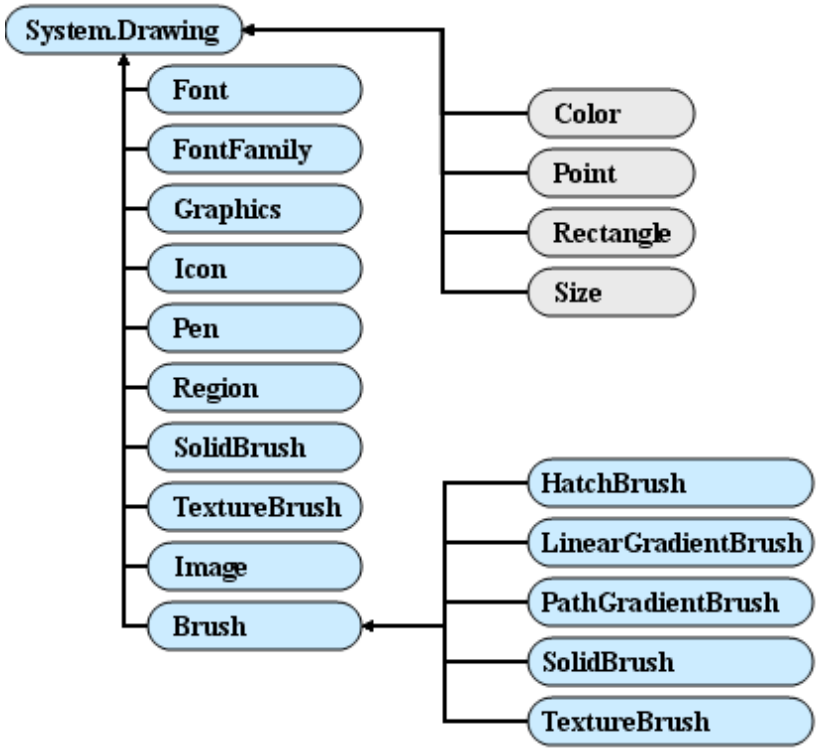
Graphics, Multimedia, and Networking

Chapter 6
Graphics, Multimedia, Networking

In compoter graphics, origin is at the left upper corner of the screen.



Here are the contents of Drawing namespace.



Color class

ARGB values, Value range from 0 to 255

Some available colors ...

Color	RGB value	**Color**	RGB value
Orange	255, 200, 0	**White**	255, 255, 255
Pink	255, 175, 175	**Gray**	128, 128, 128
Cyan	0, 255, 255	**DarkGray**	64, 64, 64
Magenta	255, 0, 255	**Red**	255, 0, 0
Yellow	255, 255, 0	**Green**	0, 255, 0
Black	0, 0, 0	**Blue**	0, 0, 255

Important static Methods of the Color

FromArgb	Creates a color based on red, green and blue values expressed as ints from 0 to 255. Overloaded version allows specification of alpha, red, green and blue.
FromName	Creates a color from a name, passed as a string.

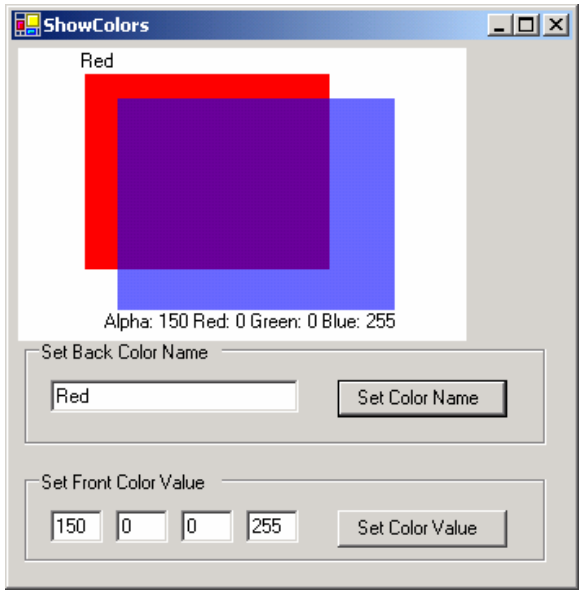
Classes that derive from class **Brush**.

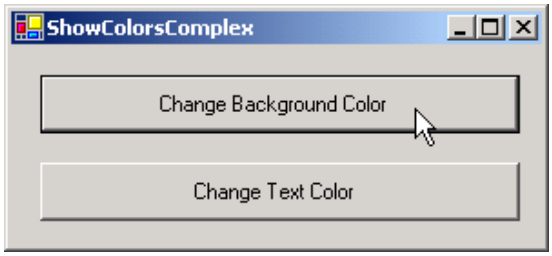
HatchBrush	Uses a rectangular brush to fill a region with a pattern. The pattern is defined by a member of the **HatchStyle** enumeration, a foreground color and a background color.
LinearGradientBrush	Fills a region with a gradual blend of one color into another. Linear gradients are defined along a line. They can be specified by the two colors, the angle of the gradient and either a rectangle or two points.
SolidBrush	Fills a region with one color. Defined by a **Color** object.
TextureBrush	Fills a region by repeating a specified **Image** across the surface.

```
1  // ShowColors.cs  :  Using different colors in C#.
4  using System;
5  using System.Drawing;
6  using System.Collections;
```

```
7   using System.ComponentModel;
8   using System.Windows.Forms;
9   using System.Data;
11
13   class ShowColors : System.Windows.Forms.Form
14   {
21  private System.Windows.Forms.TextBox colorNameTextBox;
22     private System.Windows.Forms.TextBox alphaTextBox;
23     private System.Windows.Forms.TextBox redTextBox;
24     private System.Windows.Forms.TextBox greenTextBox;
25     private System.Windows.Forms.TextBox blueTextBox;
26     private System.Windows.Forms.Button colorValueButton;
27     private System.Windows.Forms.Button colorNameButton;
28     private System.Windows.Forms.GroupBox nameGroup;
29   private System.Windows.Forms.GroupBox colorValueGroup;
31     Color behindColor = Color.Wheat;
32     // color for back rectangle
33     Color frontColor = Color.FromArgb( 100, 0 , 0, 255 );
34    // color for front rectangle
39
42     protected override void OnPaint( PaintEventArgs e )
43     {
44        Graphics graphicsObject = e.Graphics; // get graphics
47        SolidBrush textBrush = new SolidBrush( Color.Black );
50        SolidBrush brush = new SolidBrush( Color.White );
51
53        graphicsObject.FillRectangle( brush, 4, 4, 275, 180 );
55          // draw white background
56        graphicsObject.DrawString(  behindColor.Name,  this.Font,
textBrush, 40, 5 );
59
60        brush.Color = behindColor;
61       // set brush color and display back rectangle
62        graphicsObject.FillRectangle( brush, 45, 20, 150, 120 );
63
65     graphicsObject.DrawString(" Red: " + frontColor.R
66           + " Green: " + frontColor.G
```

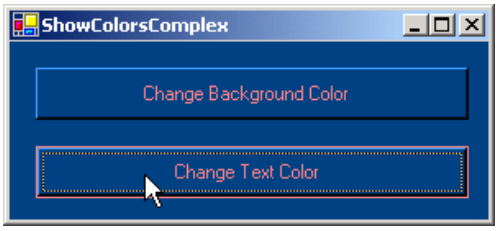
```
67      + " Blue: " + frontColor.B, this.Font, textBrush,  55, 165 );
68
69        brush.Color = frontColor;
70          // set brush color and display front rectangle
71        graphicsObject.FillRectangle( brush, 65, 35, 170, 130 );
72      }
73            private void colorNameButton_Click( object sender,
System.EventArgs e )
74      {
75      behindColor = Color.FromName( colorNameTextBox.Text );
76        Invalidate();
77      }

78    void colorValueButton_Click(object s, EventArgs e)
80      {
81        frontColor = Color.FromArgb(
82              Convert.ToInt32(alphaTextBox.Text ),
83               Convert.ToInt32( redTextBox.Text ),
84              Convert.ToInt32( greenTextBox.Text ),
85               Convert.ToInt32( blueTextBox.Text ));
88        Invalidate();// refresh Form
89      }
99 } // end class ShowColors
```






```
1   // ShowColorsComplex.cs
2   // Change background and text colors of a form.
4   using System;
5   using System.Drawing;
6   using System.Collections;
7   using System.ComponentModel;
8   using System.Windows.Forms;
9   using System.Data;
10
11              public class ShowColorsComplex : System.Windows.Forms.Form
12  {
13      private System.ComponentModel.Container components = null;
14              private System.Windows.Forms.Button backgroundColorButton;
```

```
15     private System.Windows.Forms.Button textColorButton;
16
29     private void textColorButton_Click(
30       object sender, System.EventArgs e )
31     {
32       ColorDialog colorChooser = new ColorDialog();
31
38       if (colorChooser.ShowDialog()== DialogResult.Cancel )
39                 return;
40
42       backgroundColorButton.ForeColor = colorChooser.Color;
43       textColorButton.ForeColor = colorChooser.Color;
45     } // end method textColorButton_Click
46
48  void backgroundColorButton_Click(object s, EventArgs e)
50     {
51        ColorDialog colorChooser = new ColorDialog();
56        colorChooser.FullOpen = true;
58
59        if (colorChooser.ShowDialog()== DialogResult.Cancel )
60                 return;
62
61        this.BackColor = colorChooser.Color;
63     }
64  }
```



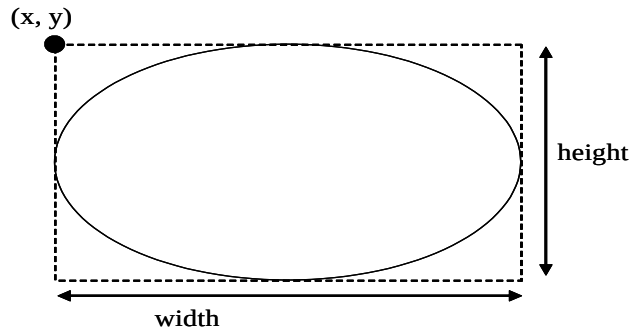
Font class

Property	Description
Bold	Tests a font for a bold font style. Returns **true** if the font is bold.
FontFamily	Represents the **FontFamily** of the **Font**.
Height	Represents the height of the font.
Italic	Tests a font for an italic font style. Returns **true** if the font is italic.
Name	Represents the font's name as a **string**.
Size	Returns a **float** value indicating the font size measured in design units.
SizeInPoints	Returns a **float** value indicating the font size measured in points.
Strikeout	Returns **true** if the font is in strikeout format.
Underline	Returns **true** if the font is underlined.

```
1   // UsingFonts.cs
2    //Demonstrating various font settings.
5     protected override void OnPaint(  PaintEventArgs paintEvent )
7     {
8        Graphics graphicsObject = paintEvent.Graphics;
9        SolidBrush brush = new SolidBrush( Color.DarkBlue );
10
11        Font arial = new Font(
12                        new FontFamily( "Arial" ), 12, FontStyle.Bold);
13        Font tNR = new Font( "Times New Roman",
14                        12, FontStyle.Regular);
15       Font cour = new Font(
16              "Courier New", 16, FontStyle.Bold | FontStyle.Italic);
17        graphicsObject.DrawString(
18              arial.Name + " 12 point bold.", arial, brush, 10, 10 );
19       graphicsObject.DrawString(
20              tNR.Name + " 12 point plain.", tNR, brush, 10, 30 );
23    graphicsObject.DrawString(
24       cour.Name + "16 point bold,italic", cour, brush, 10, 54 );
25       } // end method OnPaint
```

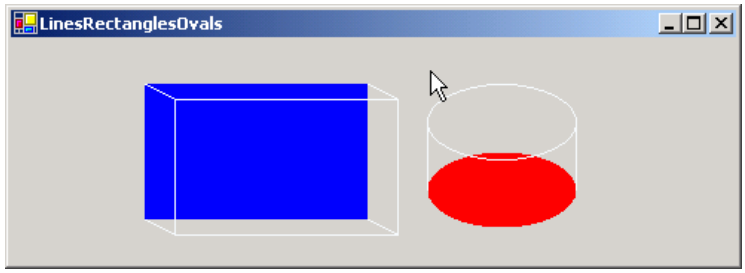
Drawing Lines, Rectangles and Ovals

- Shape outlines take **Pen**
- Solid shapes take **Brush**
- First int argument represent coordinates
- Last int argument are for width and height
- To draw circles , use **DrawEllipse** with width = height = diameter = 2* radius



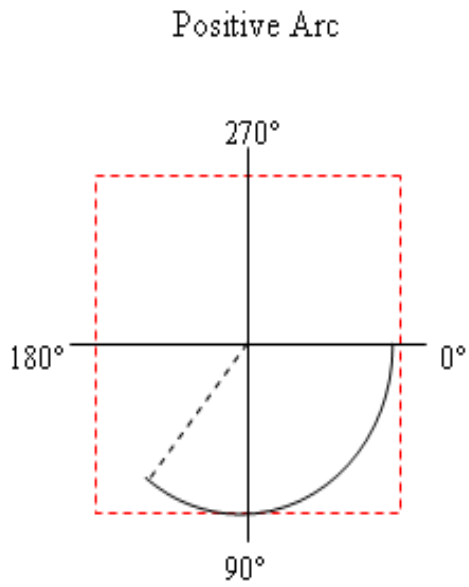
Graphics Drawing Methods

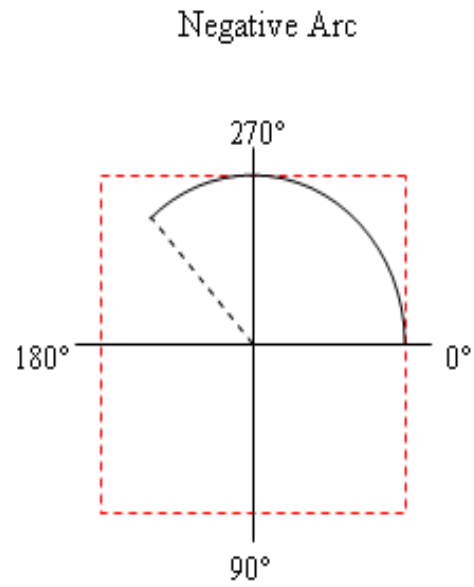
DrawLine(Pen p, int x1, int y1, int x2, int y2) Draws a line from (**x1**, **y1**) to (**x2**, **y2**). The **Pen** determines color, style and width of line.
DrawRectangle(Pen p, int x, int y, int width, int height) Draws a rectangle of the specified width and height. The top-left corner of the rectangle is at point (**x**, **y**). The **Pen** determines the color, style, and border width of the rectangle.
FillRectangle(Brush b, int x, int y, int width, int height) Similar to **FillRectangle,** but filled**.** The **Brush** determines the fill pattern inside the rectangle.
DrawEllipse(Pen p, int x, int y, int width, int height) Draws an ellipse inside a rectangle. The width and height of the rectangle are as specified, and its top-left corner is at point (**x**, **y**). The **Pen** determines color, style and border width.
FillEllipse(Brush b, int x, int y, int width, int height) Similar to **DrawEllipse** but filled




```
1   // LinesRectanglesOvals.cs:
2  //Demonstrating lines, rectangles and ovals.
4     protected override void OnPaint(  PaintEventArgs paintEvent )
5     {
6        Graphics g = paintEvent.Graphics;
7        SolidBrush brush = new SolidBrush( Color.Blue );
8        Pen pen = new Pen( Color.AliceBlue );
9
10        g.FillRectangle( brush, 90, 30, 150, 90 );  // filled rectangle
11
12       g.DrawLine( pen, 90, 30, 110, 40 );
14        g.DrawLine( pen, 90, 120, 110, 130 );
15        g.DrawLine( pen, 240, 30, 260, 40 );
16        g.DrawLine( pen, 240, 120, 260, 130 );
20
21        g.DrawRectangle( pen, 110, 40, 150, 90);
23
24        brush.Color = Color.Red;
28        g.FillEllipse( brush, 280, 75, 100, 50 ); // draw base Ellipse
29
31        g.DrawLine( pen, 380, 55, 380, 100 ); // draw lines
32        g.DrawLine( pen, 280, 55, 280, 100 );
34
35        g.DrawEllipse( pen, 280, 30, 100, 50 ); // draw Ellipse
37     } // end method OnPaint
```

Drawing Arcs





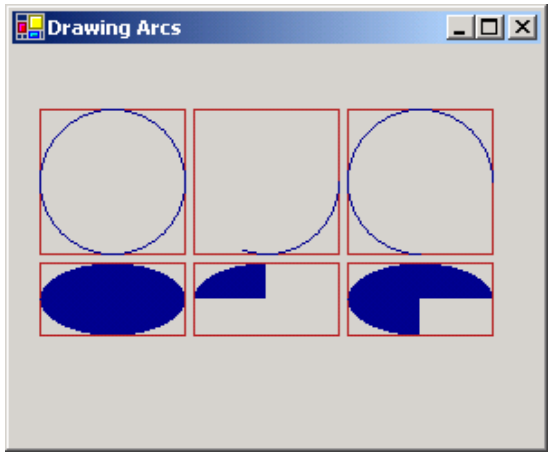
Graphics Drawing Methods

DrawArc(Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle) Draws arc of ellipse, begin from angle **startAngle** and sweeping **sweepAngle** degrees. Ellipse is defined by a bounding rectangle of width **w**, height **h** and upper-left corner (**x**,**y**).
DrawPie(Pen p, int x, int y, int width, int height, int startAngle, int sweepAngle) Functions similarly to **DrawArc**, except draws pie section of an ellipse.
FillPie(Brush b, int x, int y, int width, int height, int startAngle, int sweepAngle) Functions similarly to **DrawPie**, except draws a solid arc.

// Drawing various arcs on a form.

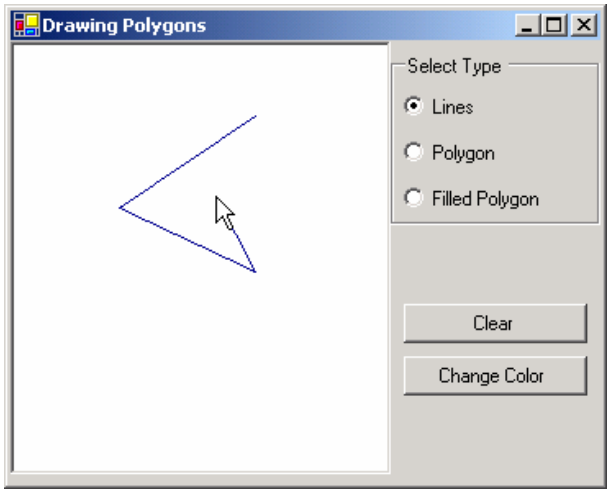
```
1                   private void Form1_Paint( object sender,
System.Windows.Forms.PaintEventArgs e )
2     {
3       Graphics g = e.Graphics;
4       Rectangle rectangle1 = new Rectangle( 15, 35, 80, 80 );
5       SolidBrush brush1 = new SolidBrush( Color.Firebrick );
6       Pen pen1 = new Pen( brush1, 1 );
7
8       SolidBrush brush2 = new SolidBrush( Color.DarkBlue );
9      Pen pen2 = new Pen( brush2, 1 );
10
11       // start at 0 and sweep 360 degrees
12       g.DrawRectangle( pen1, rectangle1 );

19     g.DrawArc( pen2, rectangle1, 0, 360 );
20
21       // start at 0 and sweep 110 degrees
22       rectangle1.Location = new Point( 100, 35 );
23       g.DrawRectangle( pen1, rectangle1 );
24     g.DrawArc( pen2, rectangle1, 0, 110 );
25
26       // start at 0 and sweep -270 degrees
27       rectangle1.Location = new Point( 185, 35 );
28       g.DrawRectangle( pen1, rectangle1 );
29     g.DrawArc( pen2, rectangle1, 0, -270 );
30
31       // start at 0 and sweep 360 degrees
32       rectangle1 = new Rectangle( 15, 120 , 80, 40 );
34       g.DrawRectangle( pen1, rectangle1 );
35      g.FillPie( brush2, rectangle1, 0, 360 );
```

```
37          // start at 270 and sweep -90 degrees
38          rectangle1.Location = new Point( 100, 120 );
39          g.DrawRectangle( pen1, rectangle1 );
40          g.FillPie( brush2, rectangle1, 270, -90 );
41
43          // start at 0 and sweep -270 degrees
44          rectangle1.Location = new Point( 185, 120 );
45          g.DrawRectangle( pen1, rectangle1 );
46          g.FillPie( brush2, rectangle1, 0, -270 );
48
49    } // end method DrawArcs_Paint
```



Drawing Polygons and Polylines

DrawLines	Draws a series of connected lines. The coordinates of each point are specified in an array of **Point**s.
DrawPolygon	Draws a polygon. The coordinates of each point are specified in an array of **Point** objects. This method draws a closed polygon, even if the last point is different from the first point.
FillPolygon	Similar to **DrawPolygon** , but draws a solid polygon.



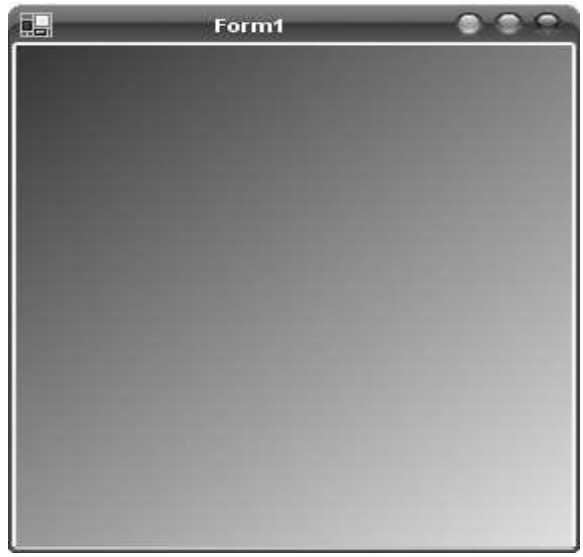

```
// DrawPolygons.cs : Demonstrating polygons.
1  public class PolygonForm : System.Windows.Forms.Form
2  {
3     private System.Windows.Forms.Button colorButton;
4     private System.Windows.Forms.Button clearButton;
5     private System.Windows.Forms.GroupBox typeGroup;
6  private System.Windows.Forms.RadioButton
filledPolygonOption;
8     private System.Windows.Forms.RadioButton lineOption;
9     private System.Windows.Forms.RadioButton polygonOption;
10    private System.Windows.Forms.Panel drawPanel;
11
16      ArrayList points = new ArrayList();
19      Pen pen = new Pen( Color.DarkBlue );
21      SolidBrush brush = new SolidBrush( Color.DarkBlue );
22
```

```
25     private void clearButton_Click(
26           object sender, System.EventArgs e )
27     {
28              points = new ArrayList();
30              drawPanel.Invalidate();
32     }
35
42     private void drawPanel_MouseDown(
41           object sender, MouseEventArgs e )
44     {
46              points.Add(new Point(e.X, e.Y));
47             drawPanel.Invalidate();
49     }
50               private void colorButton_Click(object sender,
System.EventArgs e)
51    {
52       ColorDialog dialogColor = new ColorDialog();
60          if (dialogColor.ShowDialog() == DialogResult.Cancel )
return;
61
62       pen.Color = dialogColor.Color;              brush.Color =
dialogColor.Color;
65       drawPanel.Invalidate();
67    }
81                 private void drawPanel_Paint(object sender,
PaintEventArgs e )
83     {
85              Graphics g = e.Graphics;
88              if ( points.Count > 1 )
89              {
90                     Point[] pointArray =  ( Point[] )points.ToArray(
points[ 0 ].GetType() );
94
95                      if ( polygonOption.Checked )
       g.DrawPolygon( pen, pointArray );
100                    else if ( lineOption.Checked )
       g.DrawLines( pen, pointArray );
```

```
115                 else if ( filledPolygonOption.Checked )
        g.FillPolygon( brush, pointArray );
120             }
122     }
```

In **all** **lineOption_CheckedChanged**, **polygonOption_CheckedChanged**, **filledPolygonOption_CheckedChanged** } **write** **drawPanel.Invalidate();**

Using LinearGradientBrush



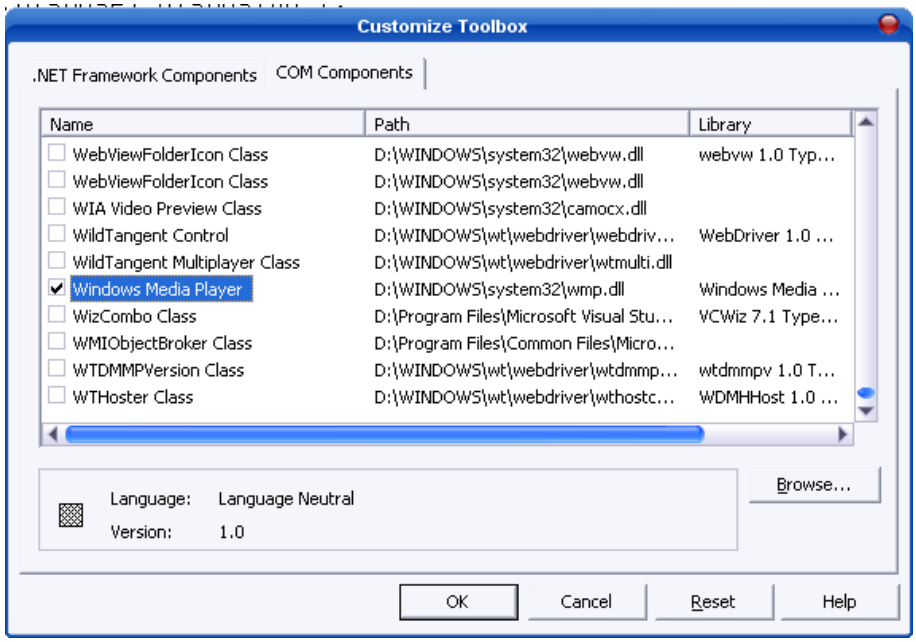

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
        System.Drawing.Drawing2D.LinearGradientBrush  gb ;
        gb                              =                               new
System.Drawing.Drawing2D.LinearGradientBrush (
new Point(0, 0),
new Point(this.Width, this.Height),
Color.Red,
Color.Yellow);
        e.Graphics.FillRectangle(gb, new Rectangle(0, 0, this.Width,
this.Height));
}
```

Animating a Series of Images

```
1   // LogoAnimator.cs :  Program that animates a series of images.
3
12  public class Form1 : System.Windows.Forms.Form
13  {
14    private System.Windows.Forms.PictureBox logoPictureBox;
15    private System.Windows.Forms.Timer Timer;
17
18    private ArrayList images = new ArrayList();
19    private int count = -1;
20
21    public Form1()
22    {
23      InitializeComponent();
24
25      for ( int i = 0; i < 30; i++ )   images.Add( Image.FromFile(
"images/deitel" + i +".gif" ) );
28
30      logoPictureBox.Image = ( Image ) images[ 0 ];
33      logoPictureBox.Size = logoPictureBox.Image.Size;
35    }
45    private void Timer_Tick(object sender, System.EventArgs e
)
47    {
49      count = ( count + 1 ) % 30;
52      logoPictureBox.Image = ( Image )images[ count ];
54    }
56  }
```

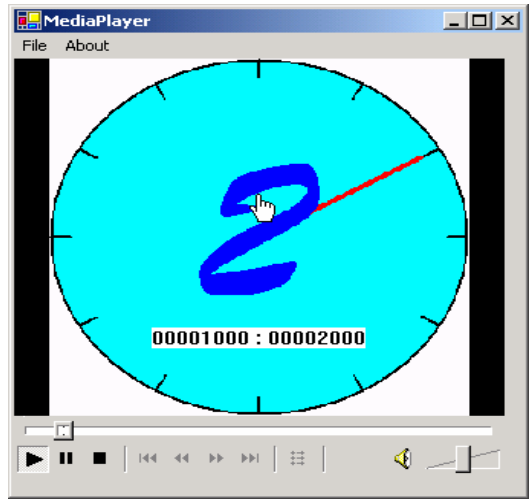
Windows Media Player

Tools Menu >> Add/Remove Toolbox Items .. >> Select "Window Media Player" >> press Ok




```
13  public class Form1 : System.Windows.Forms.Form
14  {
15                  private System.Windows.Forms.MainMenu applicationMenu;
16      private System.Windows.Forms.MenuItem fileItem;
17      private System.Windows.Forms.MenuItem openItem;
18      private System.Windows.Forms.MenuItem exitItem;
19      private System.Windows.Forms.MenuItem aboutItem;
20                  private System.Windows.Forms.MenuItem aboutMessageItem;
21              private System.Windows.Forms.OpenFileDialog openMediaFileDialog;
23      private AxMediaPlayer.AxMediaPlayer player;

37              private void openItem_Click(object sender, System.EventArgs e)
39      {
```

```
40        openMediaFileDialog.ShowDialog();
42        player.FileName = openMediaFileDialog.FileName;
43
44                player.Size = new Size( player.ImageSourceWidth,
player.ImageSourceHeight );
49                    this.Size = new Size( player.Size.Width + 20,
player.Size.Height + 60 );
50      }
51          private void aboutMessageItem_Click( object sender,
System.EventArgs e)
53      {
54       player.AboutBox();
56      }
64   private void exitItem_Click
65     (object sender, EventArgs e)
66      {
67        Application.Exit();
68      }
69   }
```

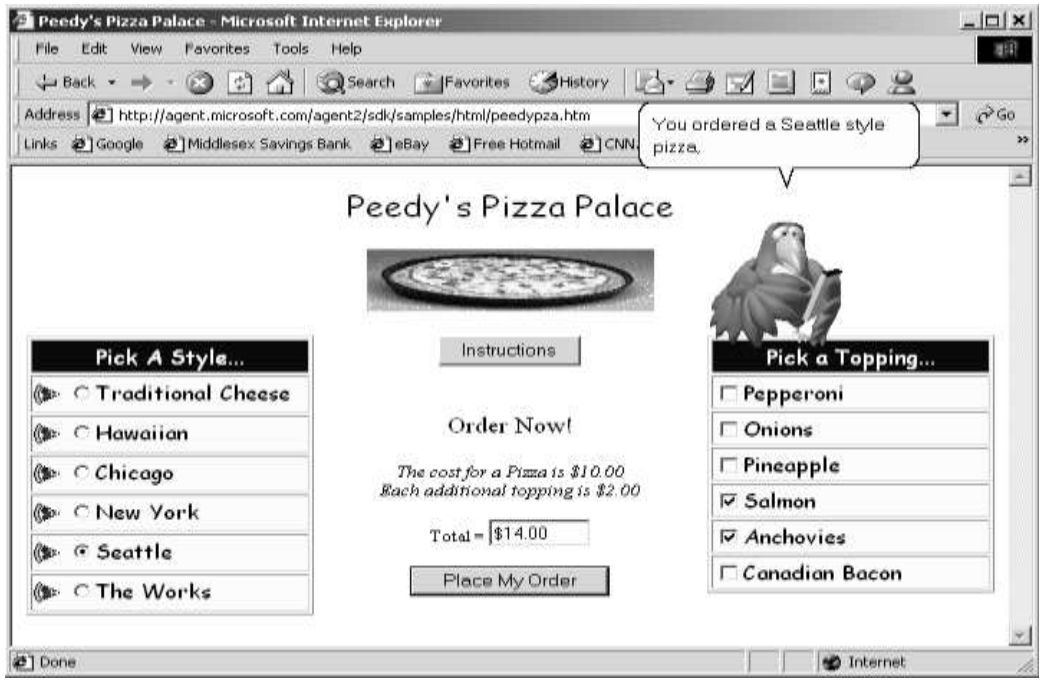


Microsoft Agent

Interactivity is essential, that aids users with questions. It has speech recognition engine plus a text-to-speech engine. Available characters are :

- **Genie**,
- **Merlin**,
- **Peedy**,
- **Robby**
- And moooore

Tools Menu >>
Add/Remove Toolbox Items>>
"Microsoft Agent" >> Ok



```
1   public class Agent : System.Windows.Forms.Form
3   {
5     private System.Windows.Forms.ComboBox actionsCombo;
6                     private     System.Windows.Forms.ComboBox
characterCombo;
8     private System.Windows.Forms.Button speakButton;
9     private System.Windows.Forms.GroupBox characterGroup;
13      private System.Windows.Forms.TextBox speechTextBox;
14      private System.Windows.Forms.TextBox locationTextBox;
20      private AxAgentObjects.AxAgent mainAgent;
30      private AgentObjects.IAgentCtlCharacter speaker;        //
current agent object
41          private void locationTextBox_KeyDown(object sender,
KeyEventArgs e )
43      {
44        if ( e.KeyCode == Keys.Enter )
45        {
47          string location = locationTextBox.Text;
50          try
51          {
53                    mainAgent.Characters.Load( "Genie", location +
"Genie.acs" );
56                   mainAgent.Characters.Load( "Merlin", location +
"Merlin.acs" );
59                    mainAgent.Characters.Load( "Peedy", location +
"Peedy.acs" );
64
67                                  locationTextBox.Enabled = false;
        speechTextBox.Enabled =  true;
68                  speakButton.Enabled = characterCombo.Enabled =
actionsCombo.Enabled = true;
72
74                  speaker = mainAgent.Characters[ "Genie" ];   // set
current character to Genie
75           GetAnimationNames();              speaker.Show( 0 );
79          }
80          catch(System.IO.FileNotFoundException ex)
```

```
81          {
82                          MessageBox.Show("Bad path", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
85          }
86        }
88     }
90              private void speakButton_Click( object sender,
System.EventArgs e )
91     {
92        if ( speechTextBox.Text == "" )       speaker.Speak("Please,
type words to speak", "" );
93        else          speaker.Speak( speechTextBox.Text, "" );
94     }
96              private void mainAgent_ClickEvent(object
s,AxAgentObjects._AgentEvents_ClickEvent e)
98     {
99        speaker.Play( "Confused" );
100       speaker.Speak( "Why are you poking me?", "" );
101       speaker.Play( "RestPose" );
103    }
116                                          private void
characterCombo_SelectedIndexChanged(object sender,
EventArgs e )
118    {
119       ChangeCharacter( characterCombo.Text );
121    }
123    private void ChangeCharacter( string name )
124    {
125       speaker.Hide( 0 );
126                   speaker = mainAgent.Characters[ name ];
GetAnimationNames();      speaker.Show( 0 );
132    }
135    private void GetAnimationNames()
136    {
138       lock( this ) // ensure thread safety
139       {
```

```
142                 IEnumerator enumerator = mainAgent.Characters[
speaker.Name ].
144                       AnimationNames.GetEnumerator();
145
149           actionsCombo.Items.Clear();
150           speaker.Commands.RemoveAll();
151
152          string voiceString;
153           while ( enumerator.MoveNext() )
154           {
155             voiceString = ( string )enumerator.Current;
157             voiceString = voiceString.Replace( "_", "underscore" );
159
160             actionsCombo.Items.Add( enumerator.Current );
163              speaker.Commands.Add( ( string )enumerator.Current,
enumerator.Current,
166                voiceString, true, false );
167           }
170             speaker.Commands.Add( "MoveHere", " MoveHere", "
MoveHere", true, true );
173         }
175       }
178       private void actionsCombo_SelectedIndexChanged(object
sender, EventArgs e )
180       {
181         speaker.StopAll( "Play" );
182         speaker.Play( actionsCombo.Text );
183         speaker.Play( "RestPose" );
185       }
188                       void      mainAgent_Command(object      s,
AxAgentObjects._AgentEvents_CommandEvent e )
190       {
192                     AgentObjects.IAgentCtlUserInput command =
(AgentObjects.IAgentCtlUserInput) e.userInput;
194
```

```
196                          if (command.Voice=="Peedy" ||
command.Voice=="Merlin"|| command.Voice == "Genie" )
200       {
201          ChangeCharacter( command.Voice );
return;
204       }
207       if ( command.Voice == " MoveHere" )
208       {
209          speaker.MoveTo( Convert.ToInt16( Cursor.Position.X -
60 ),
211                          Convert.ToInt16( Cursor.Position.Y - 60
), 5 );
212         return;
213       }
216       speaker.StopAll( "Play" );
217       speaker.Play( command.Name );
219     }
220  }
```

Networking technologies
Client-server relationship, Socket-based communications (Stream sockets)

Establishing a Simple Server (Using Stream Sockets) ... Five steps

1. Create an object of TcpListener
   Binds server to port number
2. Call Start method
   Begin connection request
3. Make connection between server and client
   Returns a Socket object
4. Processing phase
   Methods Receive and Send
5. Connection-termination phase
   Method Close

Establishing a Simple Client (Using Stream Sockets)

1. Create object of TcpClient
   Method Connect
2. Get a NetworkStream
   WriteByte and ReadByte
3. Processing phase
   Client and server communicate
4. Close connection
   Method Close

```
1   // Server.cs
5   using System;
6   using System.Drawing;
7   using System.Collections;
8   using System.ComponentModel;
9   using System.Windows.Forms;
10  using System.Threading;
11  using System.Net.Sockets;
12  using System.IO;
16  public class Server : System.Windows.Forms.Form
17  {
18    private System.Windows.Forms.TextBox inputTextBox;
19    private System.Windows.Forms.TextBox displayTextBox;
20    private Socket connection;
21    private Thread readThread;
24    private NetworkStream socketStream;
25    private BinaryWriter writer;
26    private BinaryReader reader;
27
29    public Server()
30    {
31      InitializeComponent();
32
34      readThread = new Thread( new ThreadStart( RunServer ) );
35      readThread.Start();
36    }
46              protected void Server_Closing( object sender,
CancelEventArgs e )
48    {
49        System.Environment.Exit( System.Environment.ExitCode
);
50    }
```

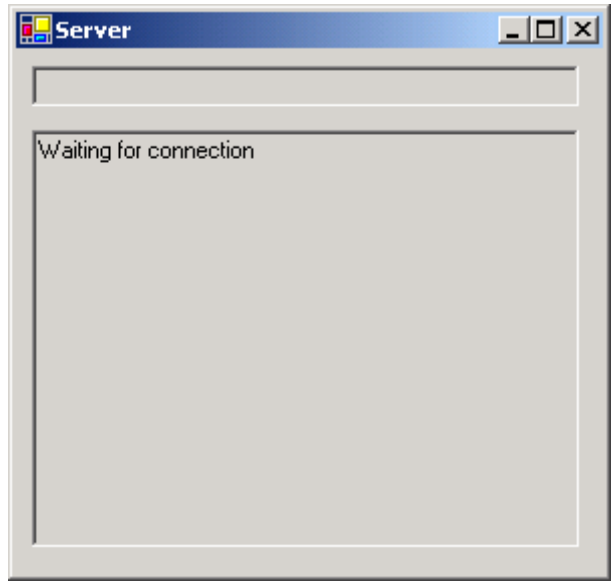
```
53        protected void inputTextBox_KeyDown( object sender,
KeyEventArgs e )
55     {
57        try
58        {
59           if ( e.KeyCode == Keys.Enter && connection != null )
60           {
61              writer.Write( "SERVER>>> " + inputTextBox.Text );
62
63                    displayTextBox.Text  +=  "\r\nSERVER>>>  "  +
inputTextBox.Text;
65
68              if ( inputTextBox.Text == "TERMINATE" )
69                 connection.Close();
70
71              inputTextBox.Clear();
72           }
73        }
74         catch ( SocketException ex) {    displayTextBox.Text +=
"\nError writing object";     }
78     }
81     public void RunServer( )  // allows client to connect & send
82     {
83        TcpListener listener;              int counter = 1;
88        try
89        {
91    listener = new TcpListener( 5000 ); // Step 1: create Listener
92
94           listener.Start(); // Step 2: waits for connection request
95
97           while ( true )    // Step 3: connection upon client request
98           {
99              displayTextBox.Text = "Waiting for connection\r\n";
100
```

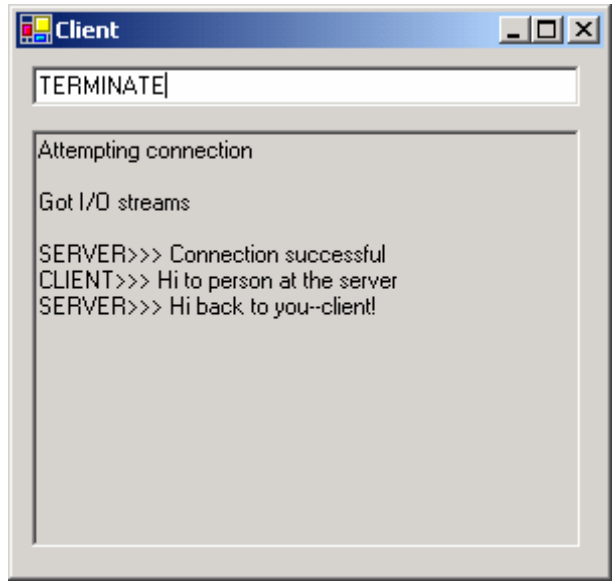
```
102             connection = listener.AcceptSocket();
105             socketStream = new NetworkStream( connection );
108             writer = new BinaryWriter( socketStream );
109             reader = new BinaryReader( socketStream );
110
111  displayTextBox.Text += "Con " + counter + " received.\r\n";
113
115             writer.Write( "SERVER>>> Connection successful" );
116
117             inputTextBox.ReadOnly = false;
118             string theReply = "";
119
121             do      // Step 4: read String data sent from client
122             {
123                     try
124                     {
126                             theReply = reader.ReadString();
129                 displayTextBox.Text += "\r\n" + theReply;
130                         }
133                     catch ( Exception ex)  {       break;         }
138               } while ( theReply != "CLIENT>>> TERMINATE"
&& connection.Connected );
140
141      displayTextBox.Text += "\r\nUser terminated connection";
143
145             inputTextBox.ReadOnly = true;
146             writer.Close();       // Step 5: close connection ↓
147             reader.Close();
148             socketStream.Close();
149             connection.Close();
151             ++counter;
152           }
153       }
155             catch ( Exception error )  {    MessageBox.Show(
error.ToString() );   }
160     } // end method RunServer
162  } // end class Server
```

```
1   // Client.cs
5   using System;
6   using System.Drawing;
7   using System.Collections;
8   using System.ComponentModel;
9   using System.Windows.Forms;
10  using System.Threading;
11  using System.Net.Sockets;
12  using System.IO;
15  public class Client : System.Windows.Forms.Form
16  {
17     private System.Windows.Forms.TextBox inputTextBox;
18     private System.Windows.Forms.TextBox displayTextBox;
20     private NetworkStream output;
21     private BinaryWriter writer;
22     private BinaryReader reader;
24     private Thread readThread;
27     private string message = "";
29
31     public Client()
32     {
33        InitializeComponent();
34
35        readThread = new Thread( new ThreadStart( RunClient ) );
36        readThread.Start();
37     }
46
47                protected void Client_Closing( object sender,
CancelEventArgs e )
49     {
50          System.Environment.Exit( System.Environment.ExitCode
);
51     }
52
54          protected void inputTextBox_KeyDown ( object sender,
KeyEventArgs e )
56     {
```

```
57        try
58        {
59          if ( e.KeyCode == Keys.Enter )
60          {
61            writer.Write( "CLIENT>>> " + inputTextBox.Text );
62
63            displayTextBox.Text    +=    "\r\nCLIENT>>>    "    +
inputTextBox.Text;
65
66            inputTextBox.Clear();
67          }
68        }
69        catch (SocketException ioe)  {
70             displayTextBox.Text += "\nError writing object";   }
74      }

77      public void RunClient()
78      {
79        TcpClient client;
82        try
83        {
84          displayTextBox.Text += "Attempting connection\r\n";
85
87          client = new TcpClient();   // Step 1: create TcpClient and
connect to server
88          client.Connect( "localhost", 5000 );
89
91          output = client.GetStream();
93      // Step 2: get NetworkStream associated with TcpClient
94          writer = new BinaryWriter( output );
95          reader = new BinaryReader( output );
96
97          displayTextBox.Text += "\r\nGot I/O streams\r\n";
98
99          inputTextBox.ReadOnly = false;
```

```
102         do
103         {
106           try     // Step 3: processing phase
107           {
109             message = reader.ReadString();
110             displayTextBox.Text += "\r\n" + message;
111           }
114       catch (Exception ex) {
115    System.Environment.Exit(System.Environment.ExitCode); }
119         } while( message != "SERVER>>> TERMINATE" );
120
121         displayTextBox.Text += "\r\nClosing connection.\r\n";
124         writer.Close(); // Step 4: close connection ↓
125         reader.Close();
126        output.Close();
127        client.Close();
127        Application.Exit();
129       }
132      catch ( Exception error )  {
133           MessageBox.Show( error.ToString() );     }
137    } // end method RunClient
139  } // end class Client
```




Client
TERMINATE
Attempting connection
Got I/O streams
SERVER>>> Connection successful
CLIENT>>> Hi to person at the server
SERVER>>> Hi back to you--client!