GDI Drawer

Fall 2019

JD Silver

GDI Drawer

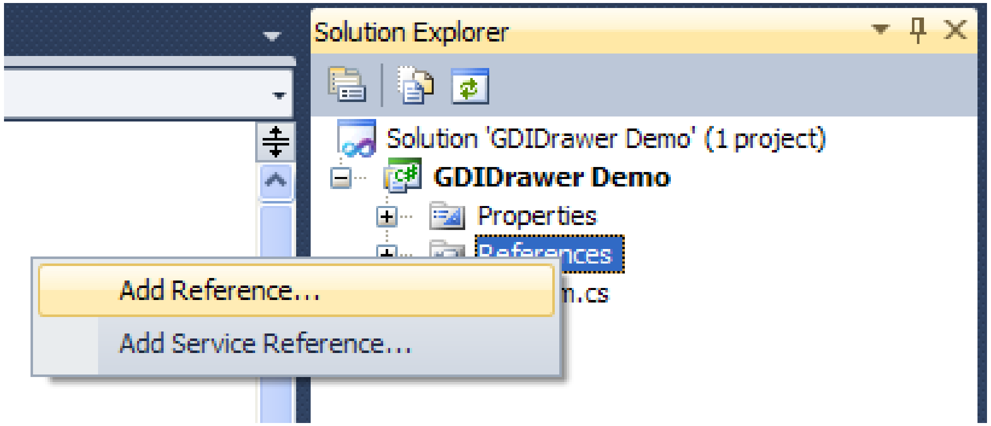
- The GDI Drawer was developed by the CNT Dept. to provide a simple graphics interface.
- It can be used to draw simple shapes and text in a separate graphics window.

Adding GDI Drawer

- The GDI Drawer must be added to the project References to use it in a program.
- Obtain a copy of GDIDrawer.dll and place it in the same folder as your project.

Adding GDI Drawer

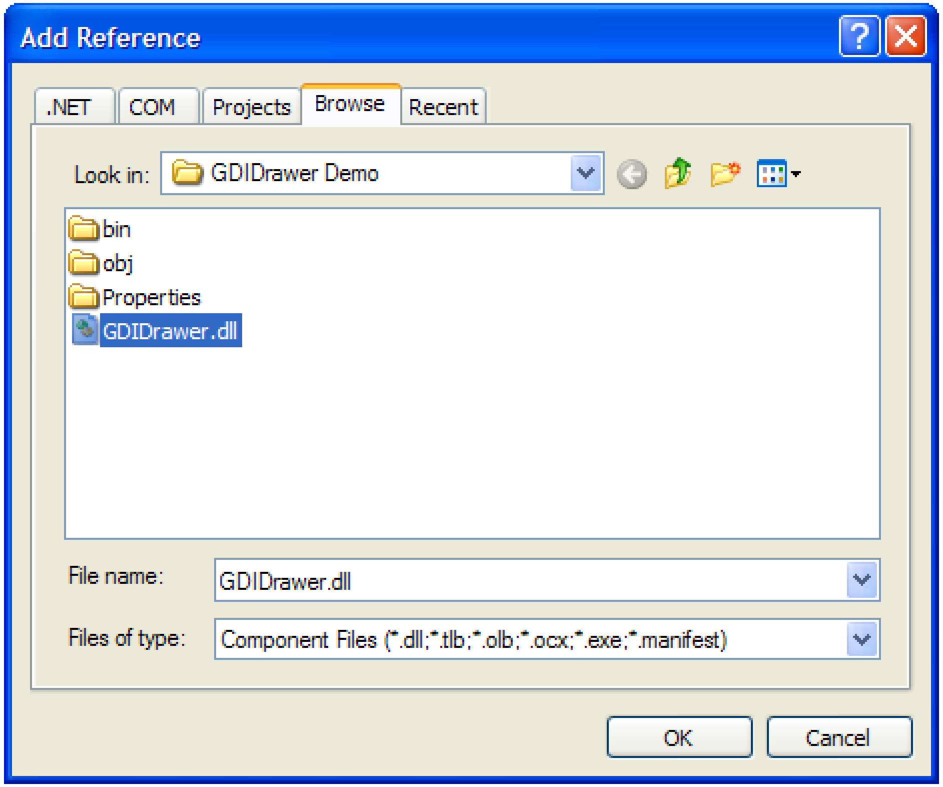
- Right-click References in the Solution Explorer and select Add Reference…



Adding GDI Drawer

- In the Add Reference dialog, select the GDIDrawer.dll file then click OK.

Adding GDI Drawer



Look in: GDIDrawer Demo

bin
obj
Properties
GDIDrawer.dll

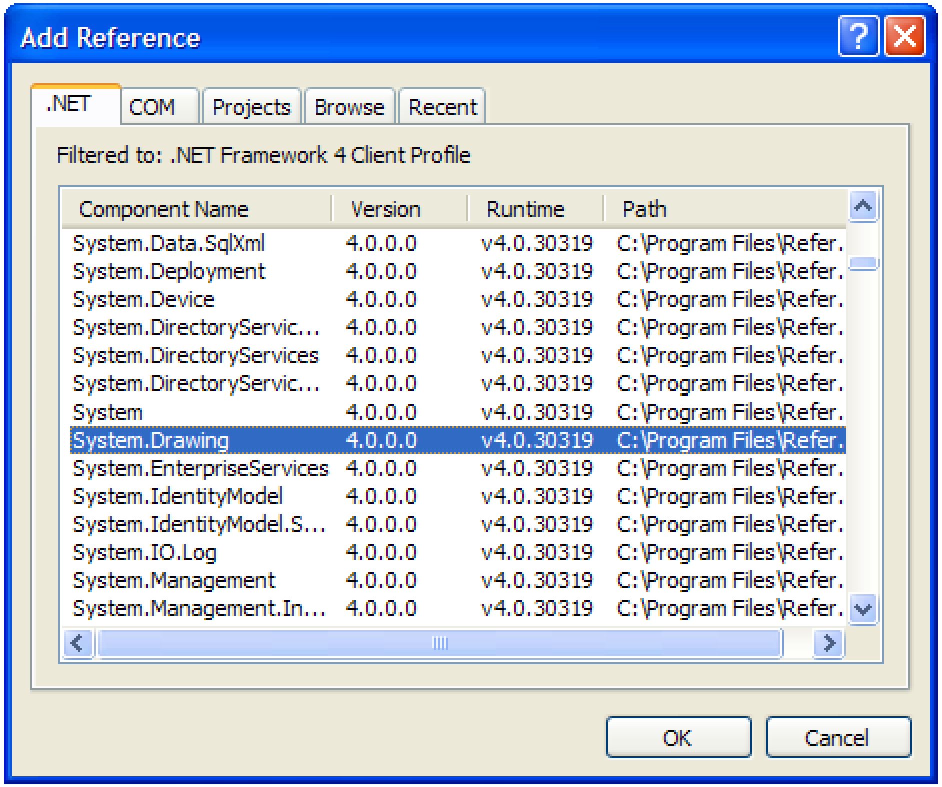
File name: GDIDrawer.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

Adding GDI Drawer

- The System.Drawing reference must also be added to the project.
- Right-click on References, select Add Reference, and the Add Reference dialog appears.
- Select the .NET tab, and pick the System.Drawing entry.

Adding GDI Drawer

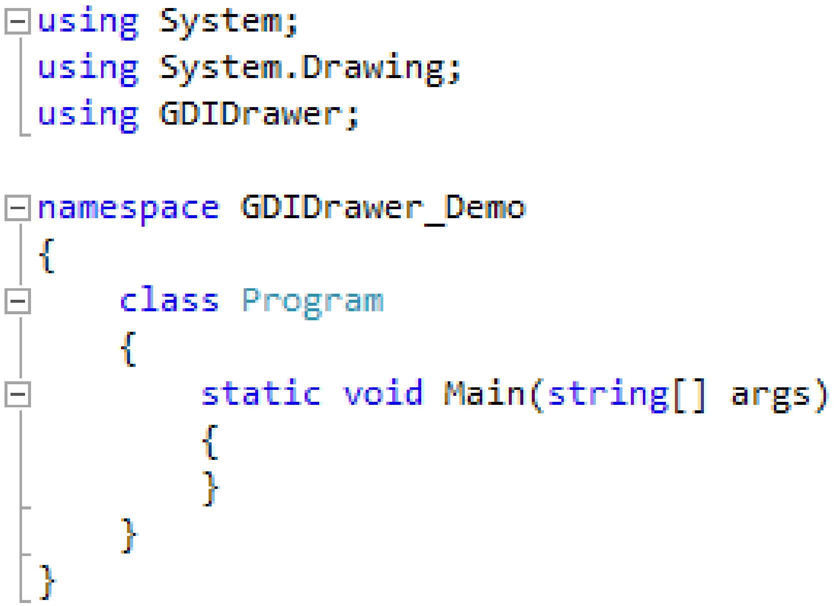


Filtered to: .NET Framework 4 Client Profile

Component Name	Version	Runtime	Path
System.Data.SqlXml	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.Deployment	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.Device	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.DirectoryServic...	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.DirectoryServices	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.DirectoryServic...	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.Drawing	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.EnterpriseServices	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.IdentityModel	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.IdentityModel.S...	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.IO.Log	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.Management	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
System.Management.In...	4.0.0.0	v4.0.30319	C:\Program Files\Refer.

Adding GDI Drawer

- Add two using statements to the program for GDIDrawer and System.Drawing.

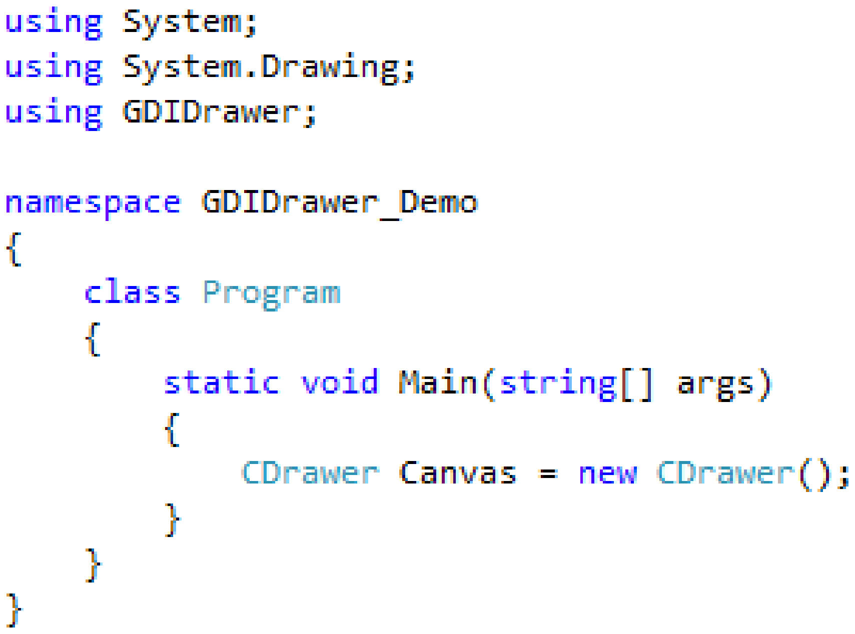


```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Creating a Drawing Window

- A drawing object and window must be created prior to performing any drawing.
- It is usually created as a CDrawer object within the main program.
- The default drawing window size is 800 pixels wide and 600 pixels high.

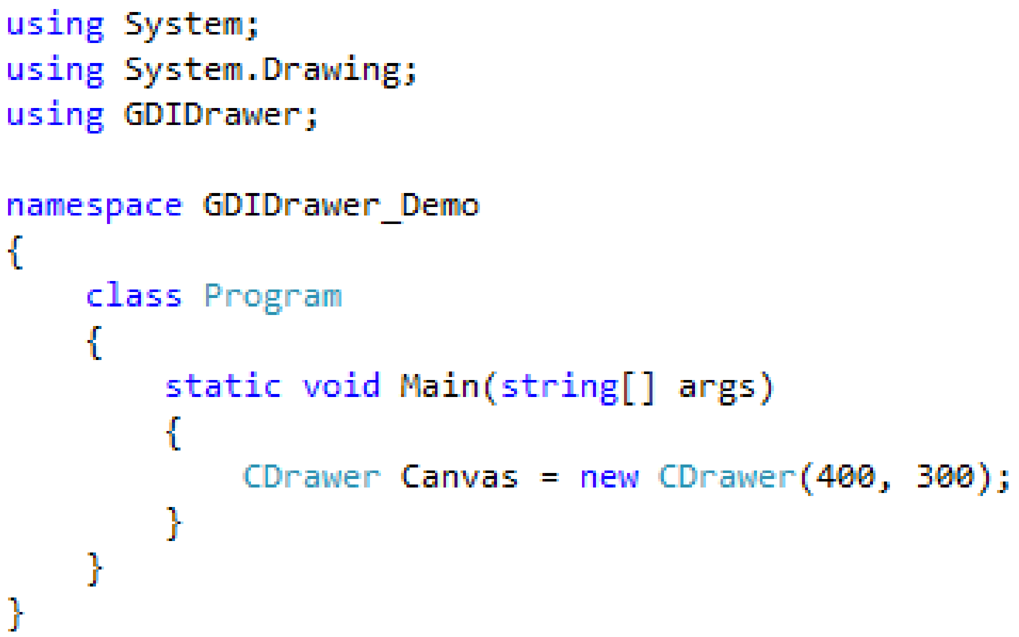
Creating a Drawing Window



```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer();
        }
    }
}
```

Creating a Drawing Window

- The size of the drawing window can also be specified as (width, height).



```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);
        }
    }
}
```

Drawing Methods

Method	Purpose
AddEllipse()	Draw an ellipse in the drawing window.
AddLine()	Draw a line in the drawing window.
AddRectangle()	Draw a rectangle in the drawing window.
AddText()	Draw text within a bounding rectangle in the drawing window.
SetBBPixel()	Draw a pixel in the backbuffer.
SetBBScaledPixel()	Draw a pixel in the backbuffer using scaled coordinates.

AddEllipse()

- There are three versions of AddEllipse()

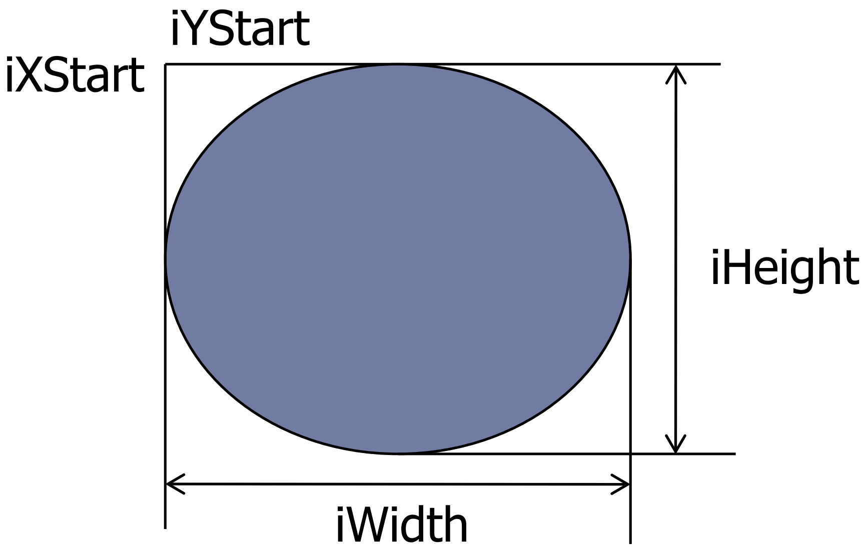
AddEllipse(int iXStart, int iYStart, int iWidth, int iHeight)

AddEllipse(int iXStart, int iYStart, int iWidth, int iHeight, Color Fill)

AddEllipse(int iXStart, int iYStart, int iWidth, int iHeight, Color Fill,
 int iBorderThickness, Color BorderColor)

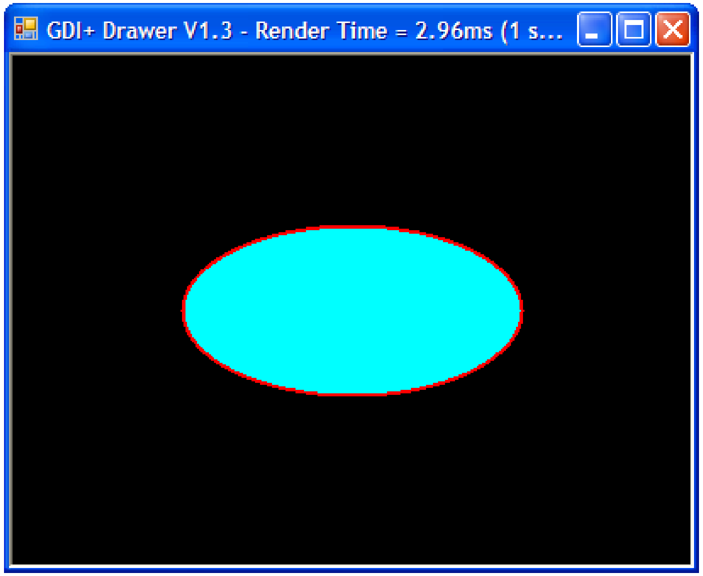
AddEllipse()

- The iXStart and iYStart values determine the upper left corner of the drawing.



AddEllipse()

```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);
            Canvas.AddEllipse(100, 100, 200, 100, Color.Aqua, 2, Color.Red);
            Console.Read();
        }
    }
}
```



AddLine()

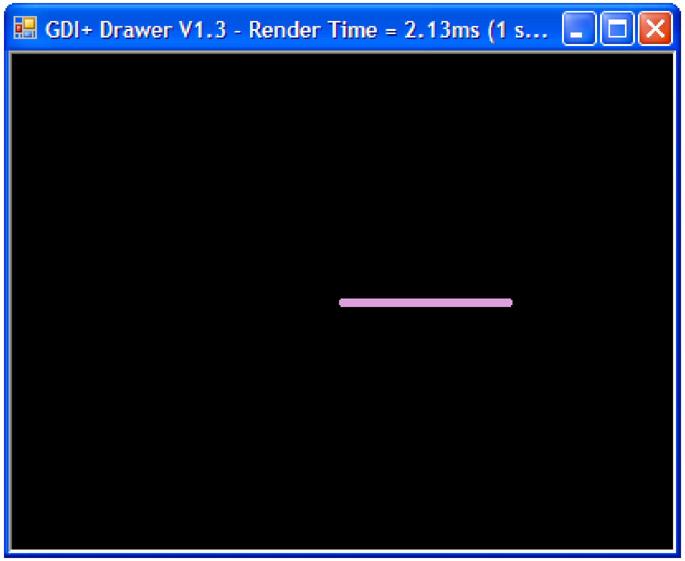
- There are four versions of AddLine()

```
AddLine(int iXStart, int iYStart, double dLength, double dRotation);

AddLine(int iXStart, int iYStart, double dLength, double dRotation,
    Color FillColor, int Thickness);

AddLine(int iXStart, int iYStart, iXEnd, iYEnd);

AddLine(int iXStart, int iYStart, iXEnd, iYEnd, Color FillColor,
    int Thickness);
```

AddLine()

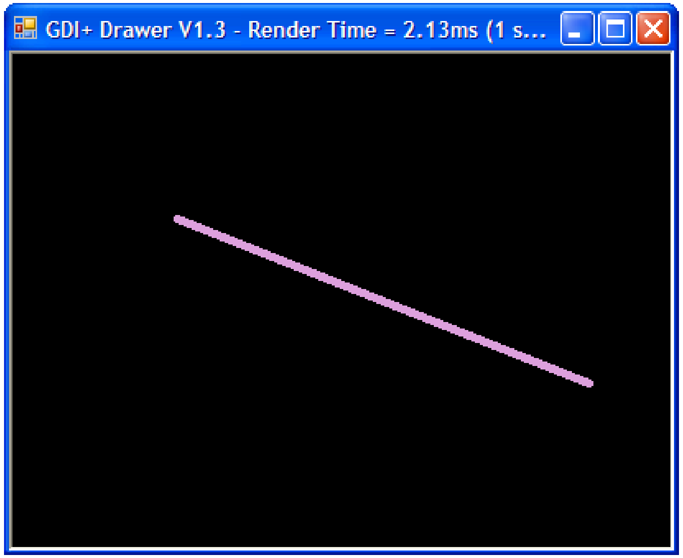
- The default fill color is grey.
- The length, start and end values are all in pixels or scaled units.
- The rotation value is in radians, with an angle of zero being the vertical, and the angle increasing in a clockwise direction.

AddLine()



```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);
            Canvas.AddLine(200, 150, 100.0, Math.PI/2, Color.Plum, 5) ;
            Console.Read();
        }
    }
}
```

AddLine()



```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);
            Canvas.AddLine(100, 100, 350, 200, Color.Plum, 5) ;
            Console.Read();
        }
    }
}
```

AddRectangle()

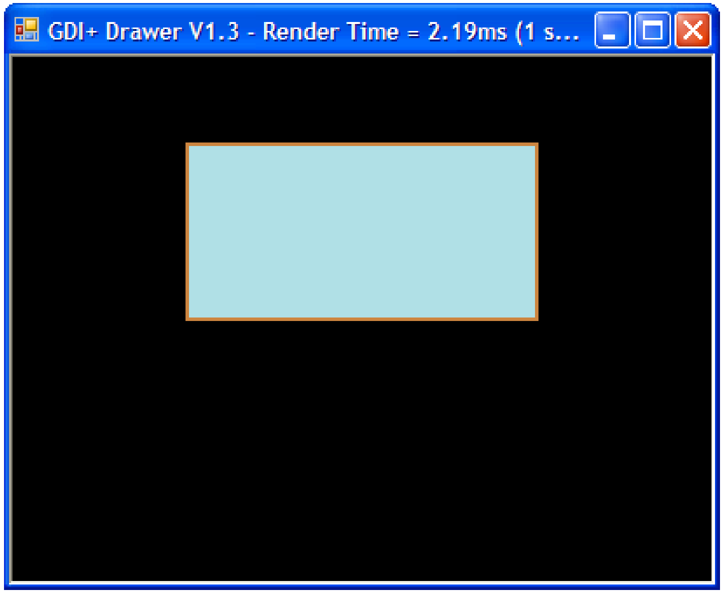
- The AddRectangle() method has 3 versions:

AddRectangle(int iXStart, int iYStart, int iWidth, int iHeight)

AddRectangle(int iXStart, int iYStart, int iWidth, int iHeight,
    Color FillColor)

AddRectangle(int iXStart, int iYStart, int iWidth, int iHeight,
    Color FillColor, int iBorderThickness, Color BorderColor)

AddRectangle()



```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);
            Canvas.AddRectangle(100, 50, 200, 100, Color.PowderBlue, 2, Color.Peru);
            Console.Read();
        }
    }
}
```

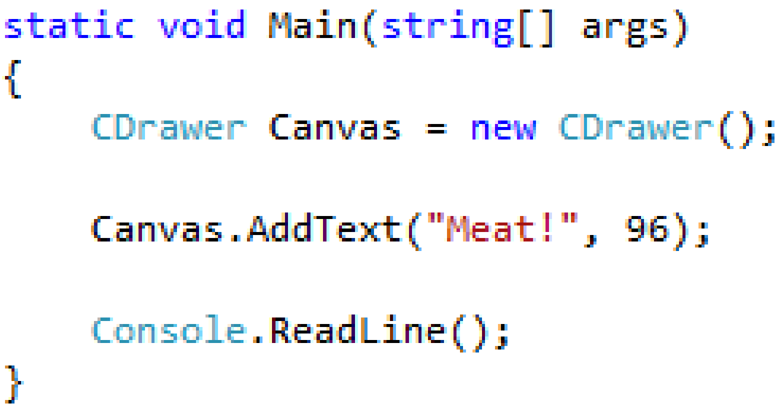
AddText()

- Adds text the the drawing window.
- There are two versions:

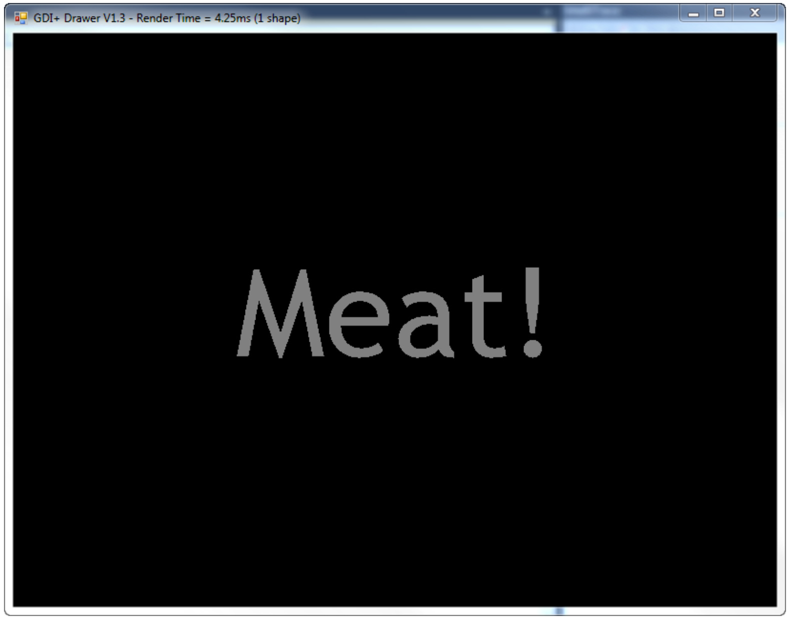
```
AddText(string sText, float fTextSize);

AddText(string sText, float fTextSize, int iXStart, int
iYStart, int iWidth, int iHeight,    Color TextColor);
```

AddText(string, float)

- Displays grey text of desired size centered in the drawing window.



```
static void Main(string[] args)
{
    CDrawer Canvas = new CDrawer();

    Canvas.AddText("Meat!", 96);

    Console.ReadLine();
}
```

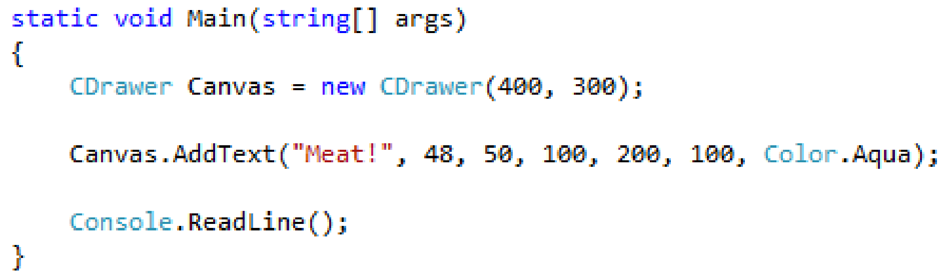


GDI+ Drawer V1.3 - Render Time = 4.25ms (1 shape)

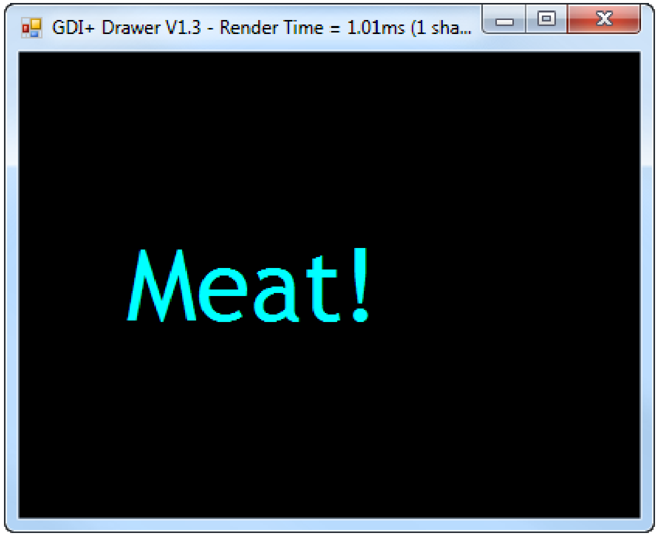
Meat!

AddText(string, float, …)

- Displays text of desired size and color centered on the rectangle specified.
- Text outside of rectangle is clipped.



```
static void Main(string[] args)
{
    CDrawer Canvas = new CDrawer(400, 300);

    Canvas.AddText("Meat!", 48, 50, 100, 200, 100, Color.Aqua);

    Console.ReadLine();
}
```



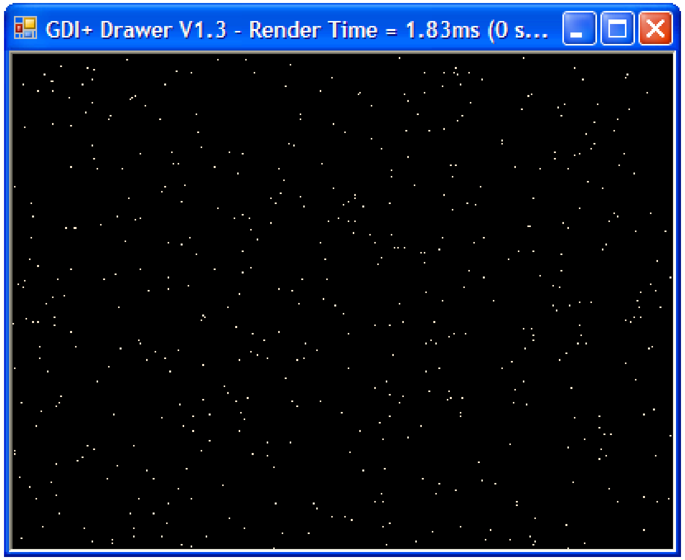
Meat!

SetBBPixel()

- SetBBPixel allows you to draw a single pixel to the backbuffer of the GDIDrawer.
- The backbuffer is **not** erased by using the Clear() method.

```
SetBBPixel(int iX, int iY, Color color)
```

SetBBPixel()



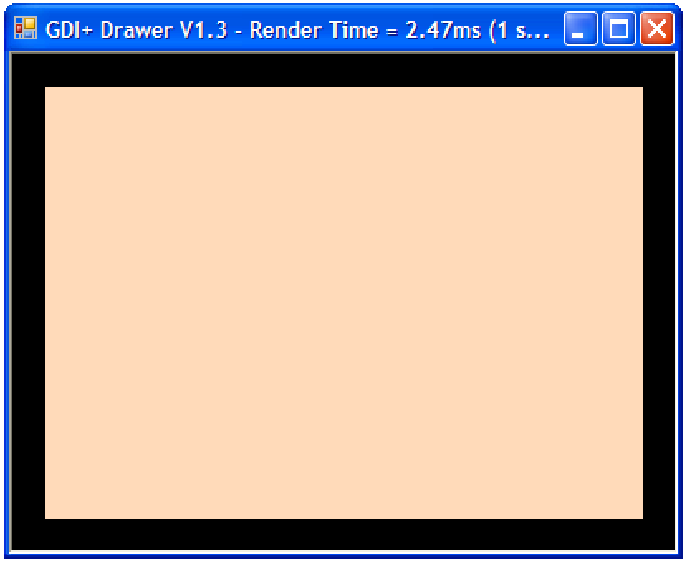
```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rndGen = new Random();
            CDrawer Canvas = new CDrawer(400, 300);

            for (int iStar = 0; iStar < 500; ++iStar)
                Canvas.SetBBPixel(rndGen.Next(400), rndGen.Next(300), Color.PapayaWhip);

            Canvas.Clear();
            Console.Read();
        }
    }
}
```

Scale

- The Scale property allows you to set the number of pixels each value of iX or iY represent in the drawing methods.
- When a shape is drawn, the values for position and size are multiplied by the scale to determine the number of pixels used.

Scale

```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            Random rndGen = new Random();
            CDrawer Canvas = new CDrawer(400, 300);

            Canvas.Scale = 20;

            Canvas.AddRectangle(1, 1, 18, 13, Color.PeachPuff);
            Console.Read();
        }
    }
}
```



Scale

- A scale of 20 with a window size of 400x300 means that the X coordinates have a maximum value of 400/20 = 20, and the Y coordinates 300/20 = 15.
- Careful use of scaling can simplify programming.

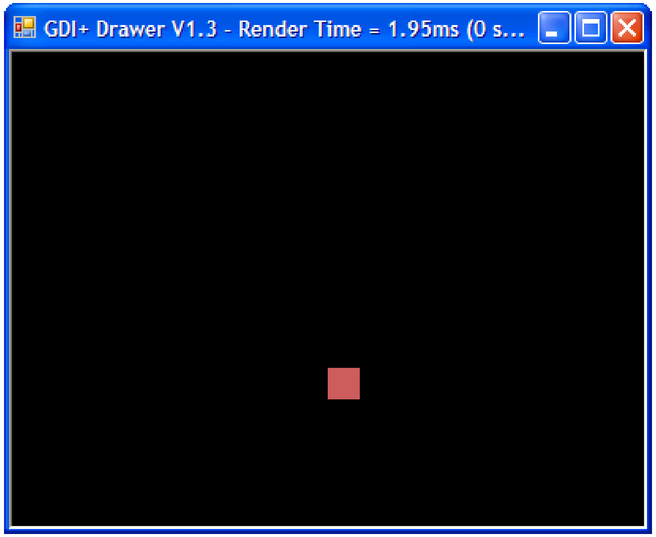
SetBBScaledPixel()

- SetBBScaledPixel() is used to draw a single pixel at a location using the current scaling factor.

  ```
  SetBBScaledPixel(int iX, int iY, Color color);
  ```

SetBBScaledPixel()

```
using System;
using System.Drawing;
using GDIDrawer;

namespace GDIDrawer_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            CDrawer Canvas = new CDrawer(400, 300);

            Canvas.Scale = 20;

            Canvas.SetBBScaledPixel(10, 10, Color.IndianRed);

            Console.Read();
        }
    }
}
```

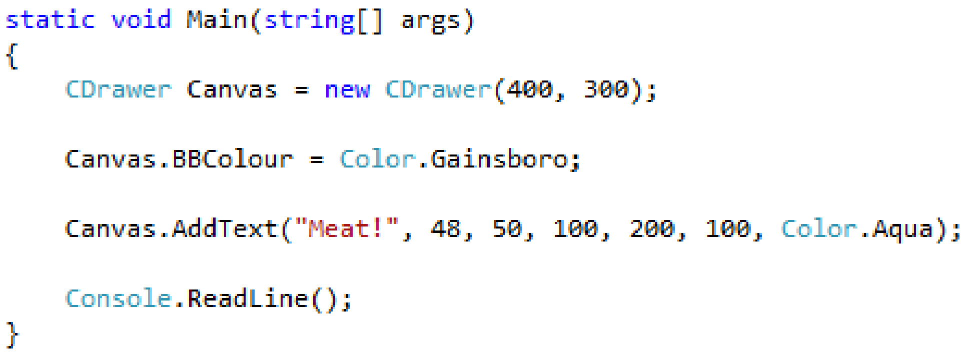


Clear()

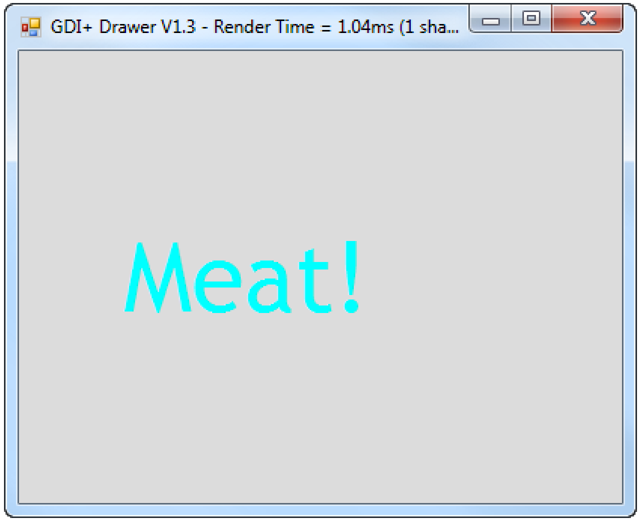
- Clears the foreground shapes from the drawer window.
- Does not clear the backbuffer.

BBColor

- A property that allows you to specify the back buffer (background) color.



```
static void Main(string[] args)
{
    CDrawer Canvas = new CDrawer(400, 300);

    Canvas.BBColour = Color.Gainsboro;

    Canvas.AddText("Meat!", 48, 50, 100, 200, 100, Color.Aqua);

    Console.ReadLine();
}
```



Meat!

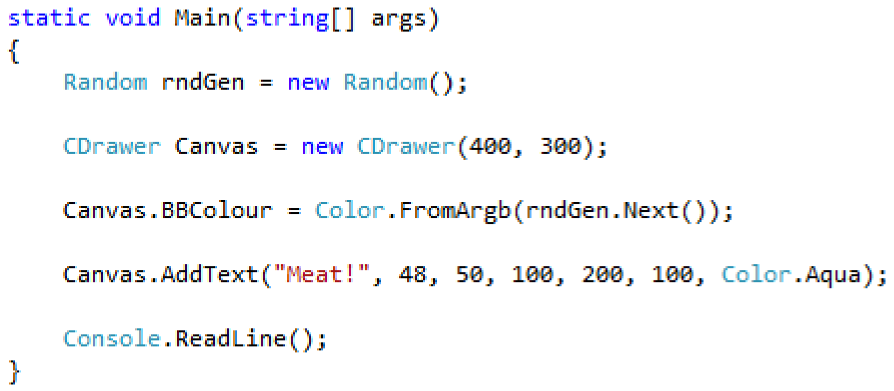
Random Colors

- A random color can be generated by use of the Color.FromArgb() method.

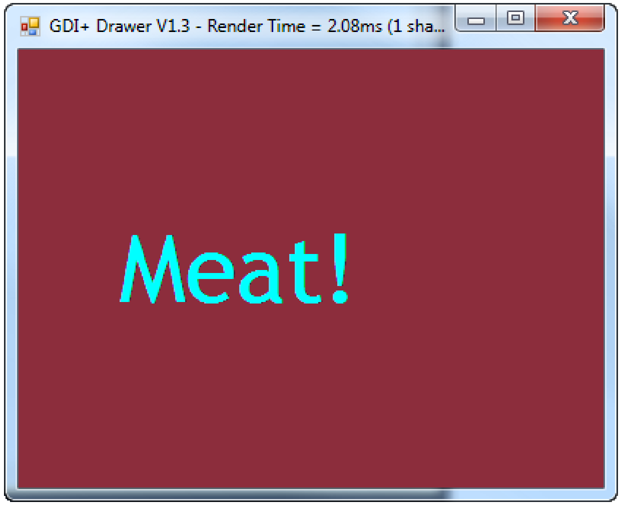
  ```
  Color.FromArgb( int32 );
  Color.FromArgb( int Alpha, Color);
  Color.FromArgb(int red, int green, int blue);
  Color.FromArgb(int Alpha, int r, int g, int b);
  ```

Random Colors

- Providing a random number to FromArgb() creates a random color.



```
static void Main(string[] args)
{
    Random rndGen = new Random();

    CDrawer Canvas = new CDrawer(400, 300);

    Canvas.BBColour = Color.FromArgb(rndGen.Next());

    Canvas.AddText("Meat!", 48, 50, 100, 200, 100, Color.Aqua);

    Console.ReadLine();
}
```



GDI+ Drawer V1.3 - Render Time = 2.08ms (1 sha...

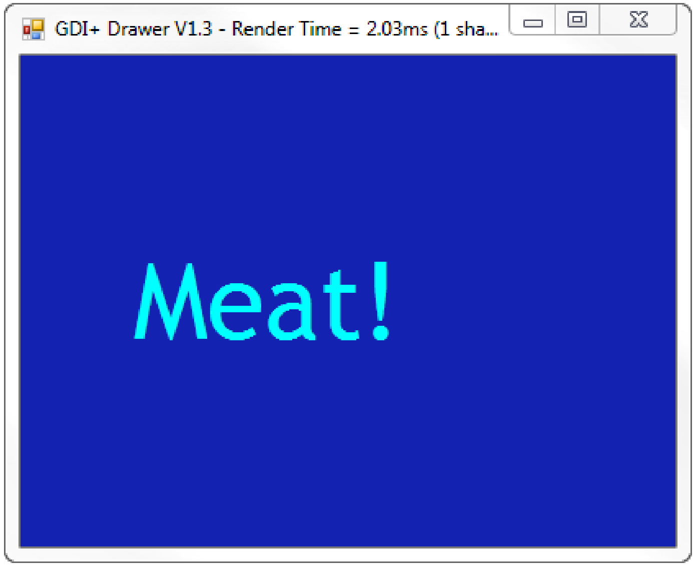
Meat!

Random Colors

- A specific color can be generated by providing values for alpha, red, green and blue.
- Each value is an 8-bit number, which may range from 0 to 255 maximum.
- Values outside the range of 0 to 255 will cause an exception to be thrown.
- The alpha value specifies the transparency of the color, with 0 being transparent, and 255 being opaque.

Random Colors

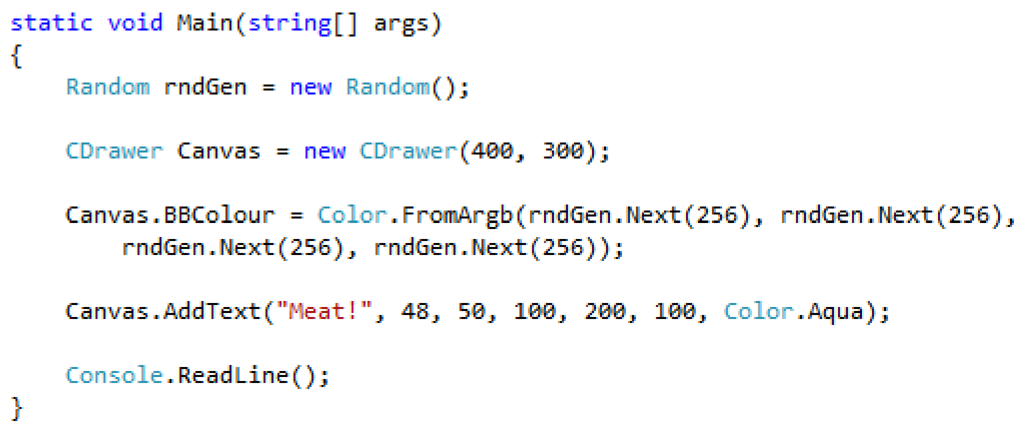
```
static void Main(string[] args)
{
    Random rndGen = new Random();

    CDrawer Canvas = new CDrawer(400, 300);

    Canvas.BBColour = Color.FromArgb(200, 20, 35, 178);

    Canvas.AddText("Meat!", 48, 50, 100, 200, 100, Color.Aqua);

    Console.ReadLine();
}
```



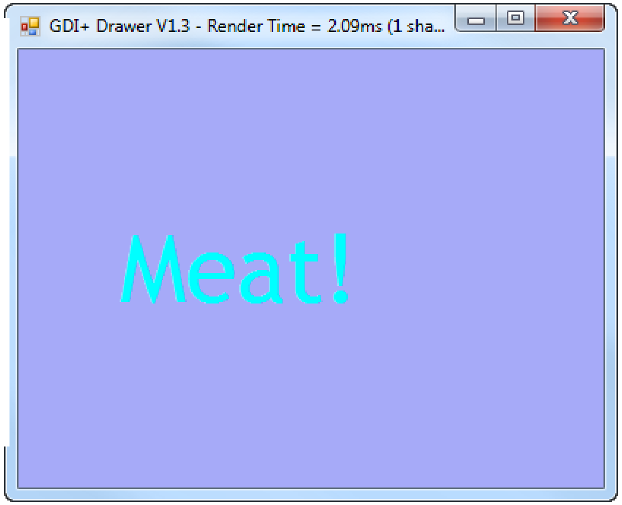
Meat!

Random Colors

- Random 8-bit values can also be used for alpha, red, green and blue separately.



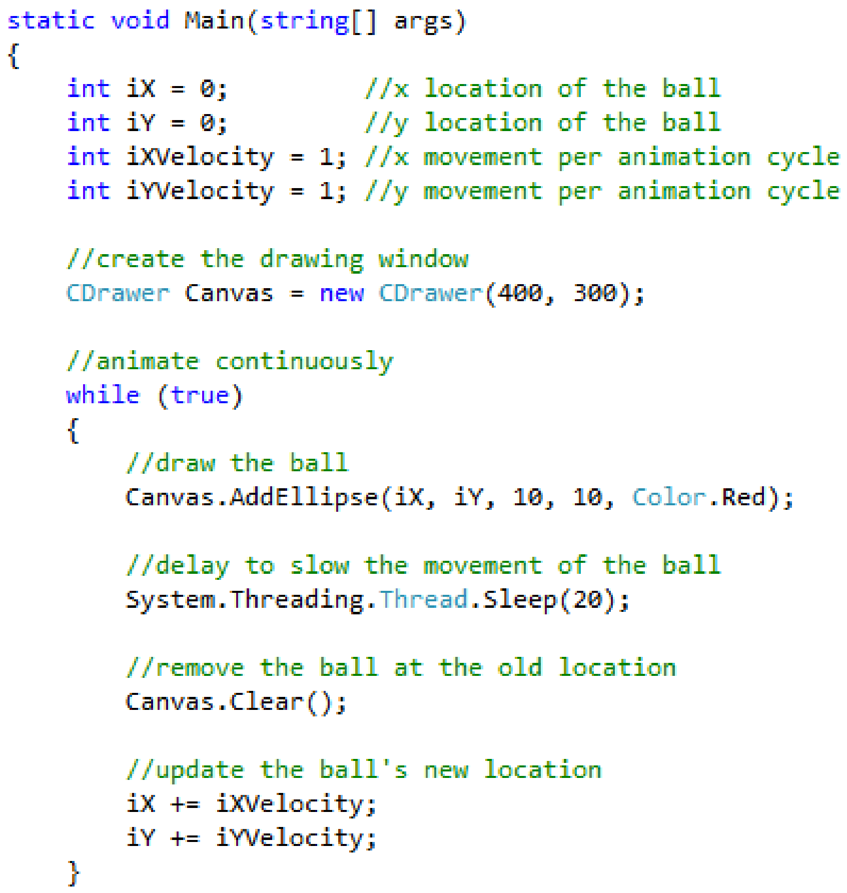
```
static void Main(string[] args)
{
    Random rndGen = new Random();

    CDrawer Canvas = new CDrawer(400, 300);

    Canvas.BBColour = Color.FromArgb(rndGen.Next(256), rndGen.Next(256),
        rndGen.Next(256), rndGen.Next(256));

    Canvas.AddText("Meat!", 48, 50, 100, 200, 100, Color.Aqua);

    Console.ReadLine();
}
```



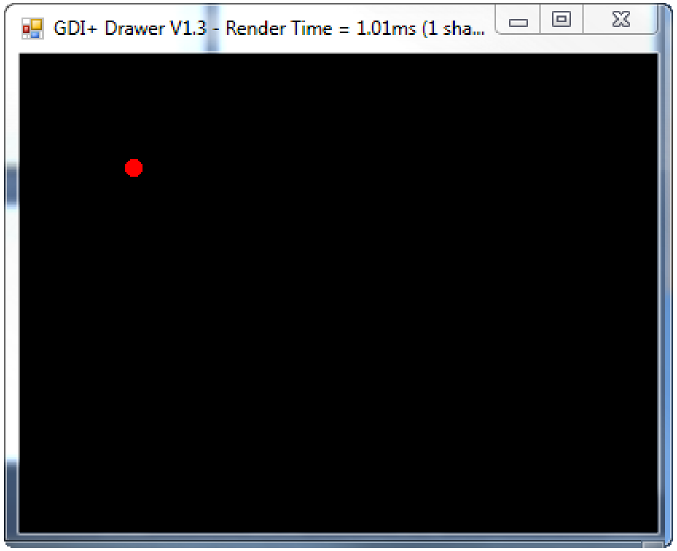
Animation

- Simple animation is possible using the GDIDrawer.
- An object can appear to move by drawing it at a new location, and erasing the old location.
- Usually a Sleep() delay is required to set the desired animation rate of the objects.
- Normally the GDIDrawer will draw each object as it is added.

Animation



```
static void Main(string[] args)
{
    int iX = 0;         //x location of the ball
    int iY = 0;         //y location of the ball
    int iXVelocity = 1; //x movement per animation cycle
    int iYVelocity = 1; //y movement per animation cycle

    //create the drawing window
    CDrawer Canvas = new CDrawer(400, 300);

    //animate continuously
    while (true)
    {
        //draw the ball
        Canvas.AddEllipse(iX, iY, 10, 10, Color.Red);

        //delay to slow the movement of the ball
        System.Threading.Thread.Sleep(20);

        //remove the ball at the old location
        Canvas.Clear();

        //update the ball's new location
        iX += iXVelocity;
        iY += iYVelocity;
    }
```

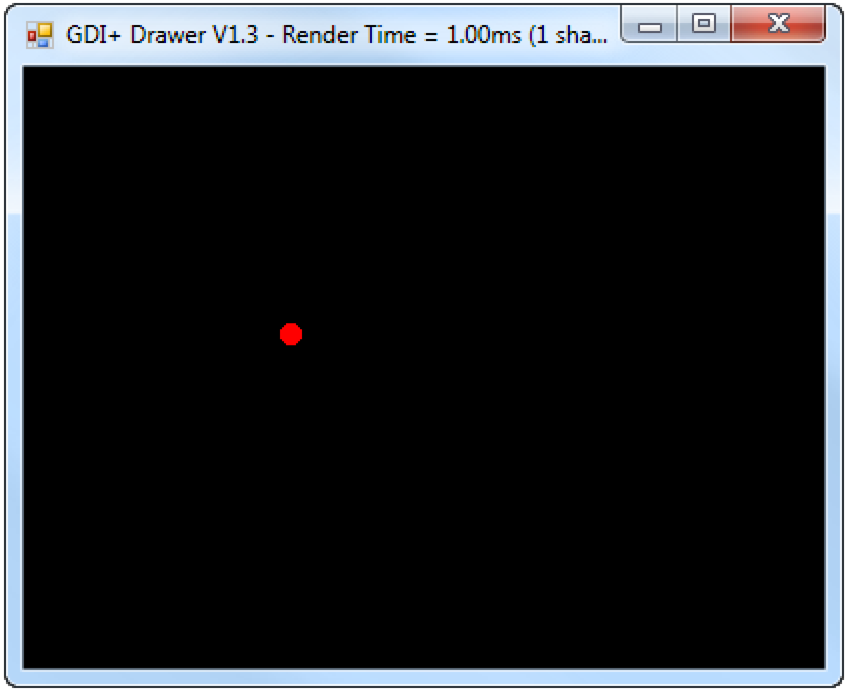


ContinuousUpdate

- The GDIDrawer will normally draw each object as it is created.
- When animating, this can lead to flicker.
- If you set ContinuousUpdate to false, then you can draw the objects in code, and use Render() to have them appear in the window.
- When many objects are being drawn, this will reduce flicker and improve the animation speed.

ContinuousUpdate

```
//create the drawing window
CDrawer Canvas = new CDrawer(400, 300);

Canvas.ContinuousUpdate = false;

//animate continuously
while (true)
{
    //draw the ball
    Canvas.AddEllipse(iX, iY, 10, 10, Color.Red);

    //render the ball to the screen
    Canvas.Render();

    //delay to slow the movement of the ball
    System.Threading.Thread.Sleep(20);

    //remove the ball at the old location
    Canvas.Clear();

    //update the ball's new location
    iX += iXVelocity;
    iY += iYVelocity;
}
```


GDI+ Drawer V1.3 - Render Time = 1.00ms (1 sha...

Render()

- Draws all objects that have been added to the GDIDrawer window.
- Render() is used to draw the objects when ContinuousUpdate is set to false.

Mouse

- The mouse location and the use of the mouse buttons can be used with the GDIDrawer.
- The mouse methods will return the mouse location as a Point object.
- The Point object contains the X and Y location of the mouse.

Mouse Methods

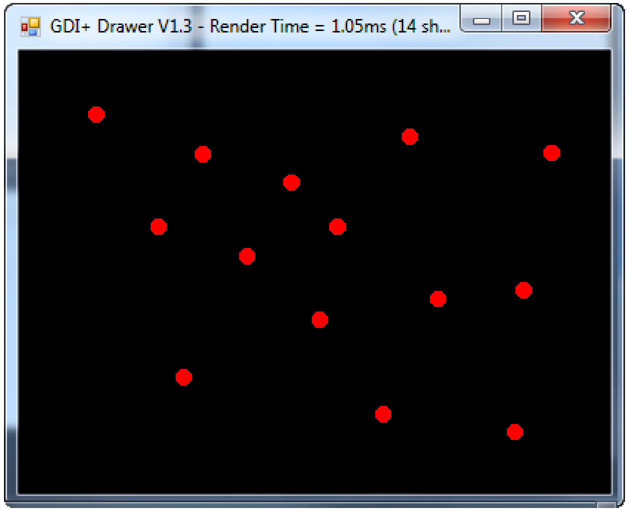
Method	Purpose
GetLastMousePostion(out Point)	Get position.
GetLastMousePostionScaled(out Point)	Get position using the scale.
GetLastMouseLeftClick(out Point)	Get the location of a left click.
GetLastMouseLeftClickScaled(out Point)	Get the location of a left click scaled.
GetLastMouseRightClick(out Point)	Get the location of a right click.
GetLastMouseRightClickScaled(out Point)	Get the location of a right click scaled.

Mouse Methods

- The mouse methods all return a true value indicating if the mouse button was pressed (true) or not (false).
- The location is returned as a Point, using an out parameter.
- The Point object contains X and Y values as integers.
- The following example uses the left mouse button to draw balls until the right mouse button is pressed.

Mouse Methods

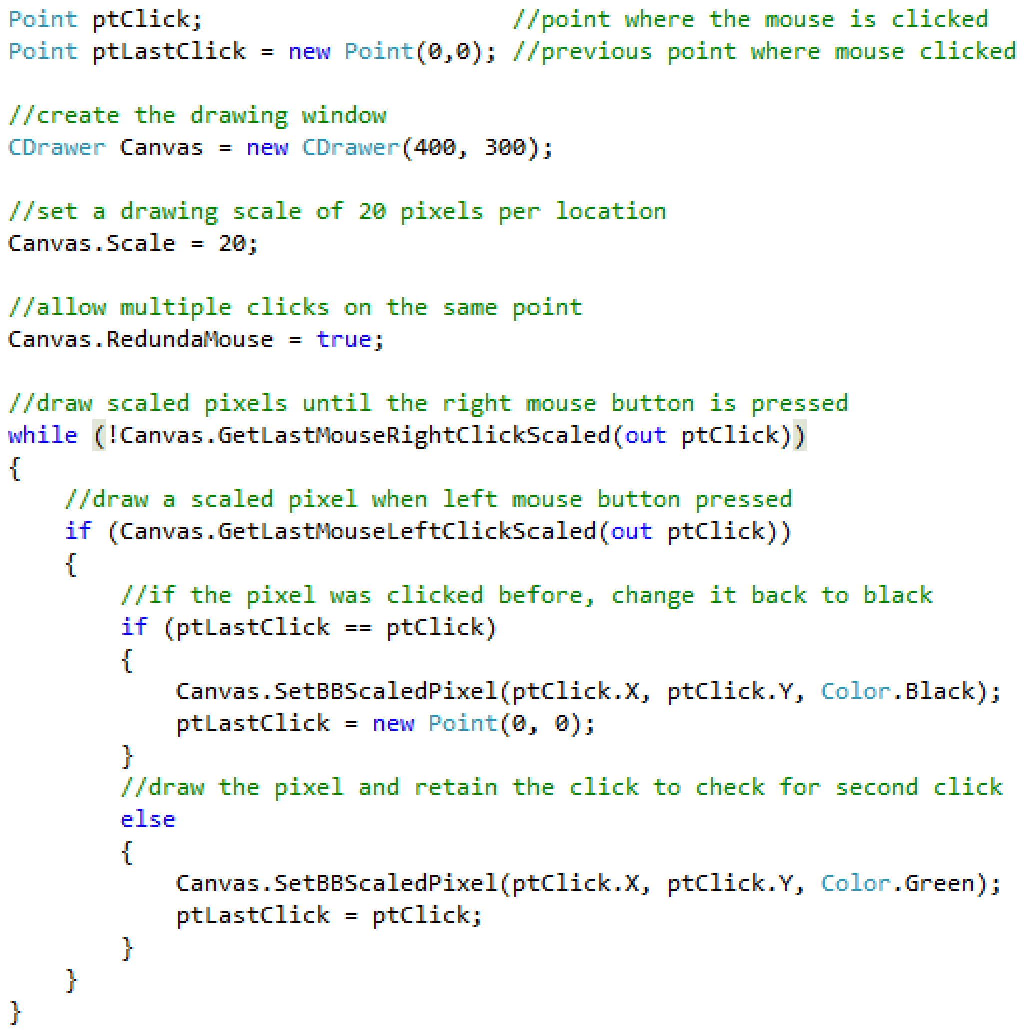
```
static void Main(string[] args)
{
    Point ptClick; //point where the mouse is clicked

    //create the drawing window
    CDrawer Canvas = new CDrawer(400, 300);

    //draw shapes only when Render is called
    Canvas.ContinuousUpdate = false;

    //draw balls until the right mouse button is pressed
    while (!Canvas.GetLastMouseRightClick(out ptClick))
    {
        //draw a ball when left mouse button pressed
        if (Canvas.GetLastMouseLeftClick(out ptClick))
        {
            //draw the ball at the mouse click position
            Canvas.AddEllipse(ptClick.X, ptClick.Y, 10, 10, Color.Red);

            //render the ball to the screen
            Canvas.Render();
        }
    }
}
```



RedundaMouse

- The GDIDrawer normally does not allow multiple mouse clicks to render at the same Point.
- This can become an issue when a scale is used which will create large sized points.
- Setting RedundaMouse to true will allow multiple clicks on the same scaled point to be registered.

RedundaMouse



```
Point ptClick;                      //point where the mouse is clicked
Point ptLastClick = new Point(0,0); //previous point where mouse clicked

//create the drawing window
CDrawer Canvas = new CDrawer(400, 300);

//set a drawing scale of 20 pixels per location
Canvas.Scale = 20;

//allow multiple clicks on the same point
Canvas.RedundaMouse = true;

//draw scaled pixels until the right mouse button is pressed
while (!Canvas.GetLastMouseRightClickScaled(out ptClick))
{
    //draw a scaled pixel when left mouse button pressed
    if (Canvas.GetLastMouseLeftClickScaled(out ptClick))
    {
        //if the pixel was clicked before, change it back to black
        if (ptLastClick == ptClick)
        {
            Canvas.SetBBScaledPixel(ptClick.X, ptClick.Y, Color.Black);
            ptLastClick = new Point(0, 0);
        }
        //draw the pixel and retain the click to check for second click
        else
        {
            Canvas.SetBBScaledPixel(ptClick.X, ptClick.Y, Color.Green);
            ptLastClick = ptClick;
        }
    }
}
```