C#の基本
～描画の方法～

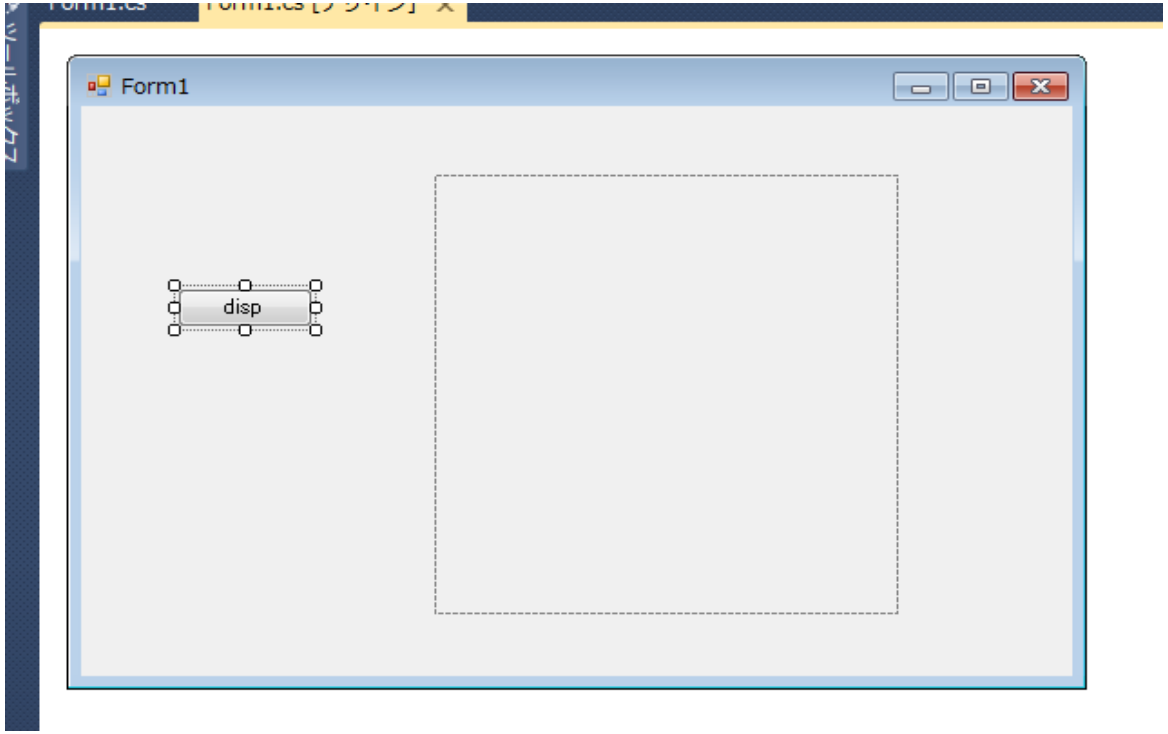
今回学ぶ事

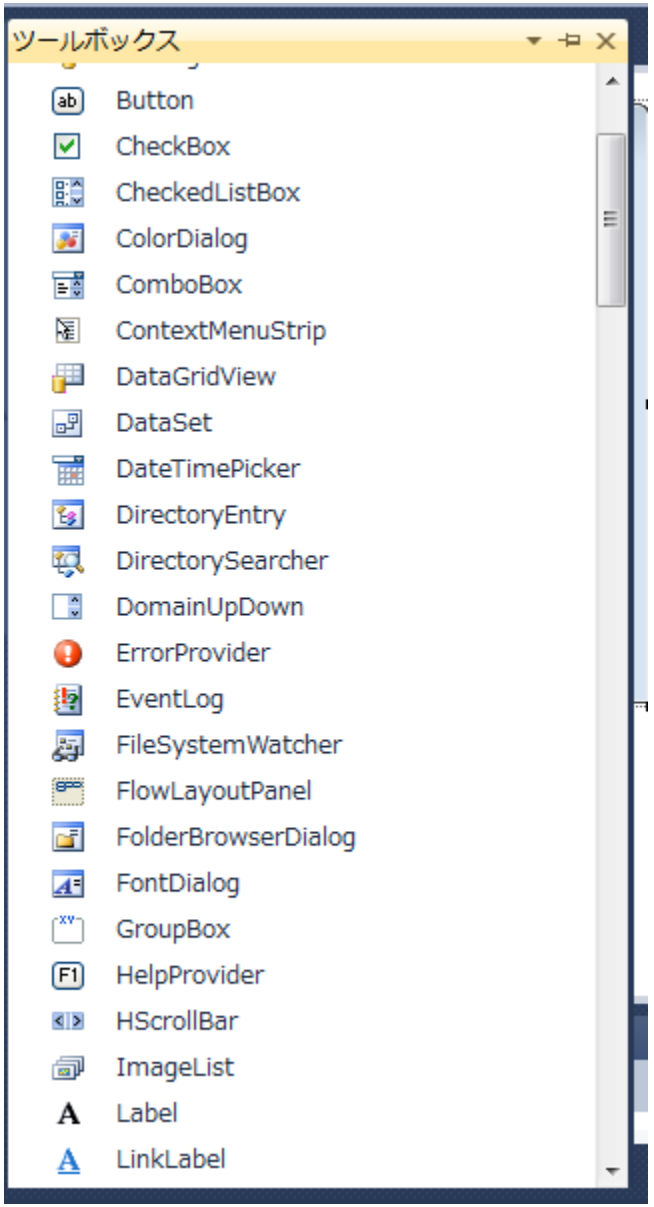
- 今回は描画において必要となるBitmapクラス、Graphicクラス、Color構造体等について解説する。

- また、PictureBoxクラスに関しても解説を行う。

フォーム作り

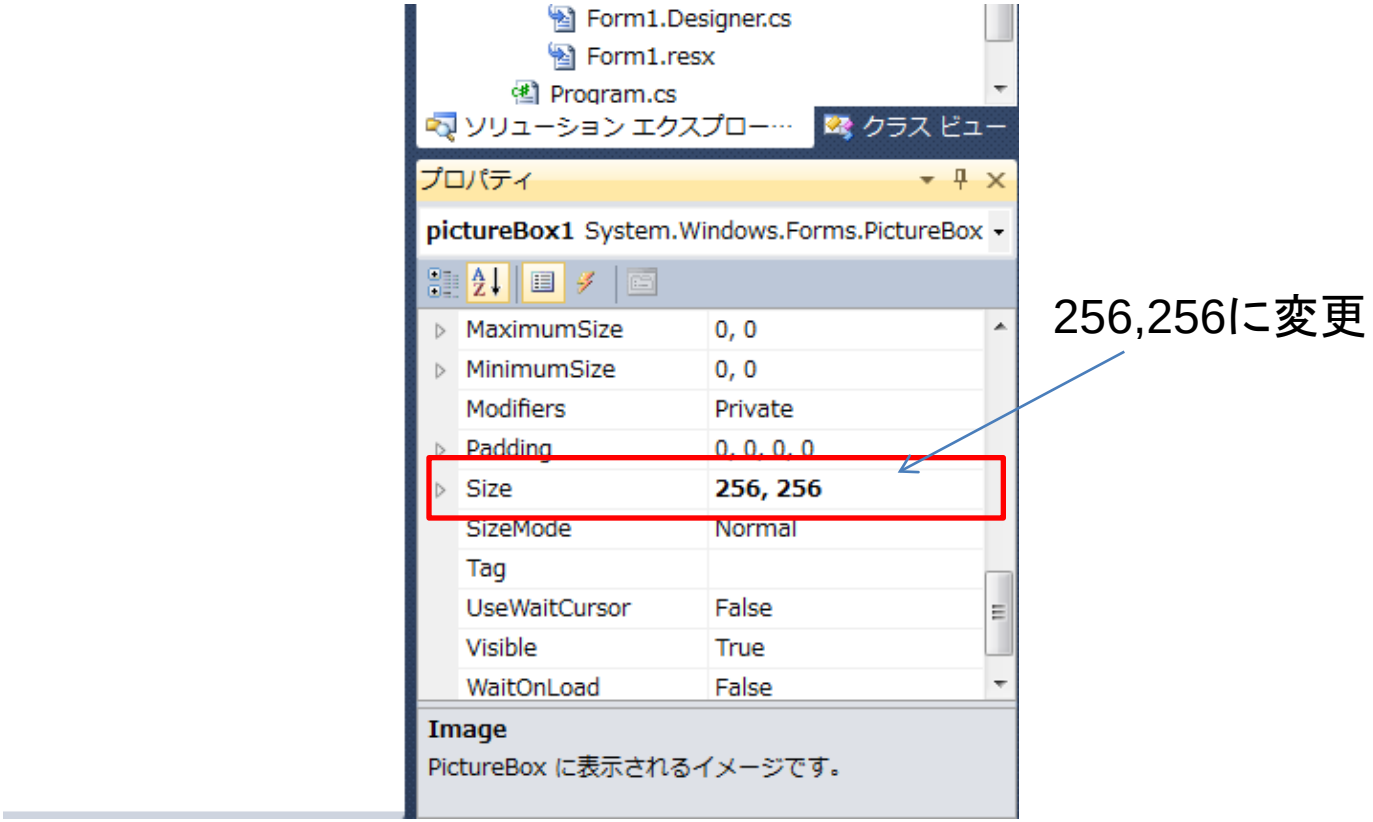
- まず、label、pictureBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



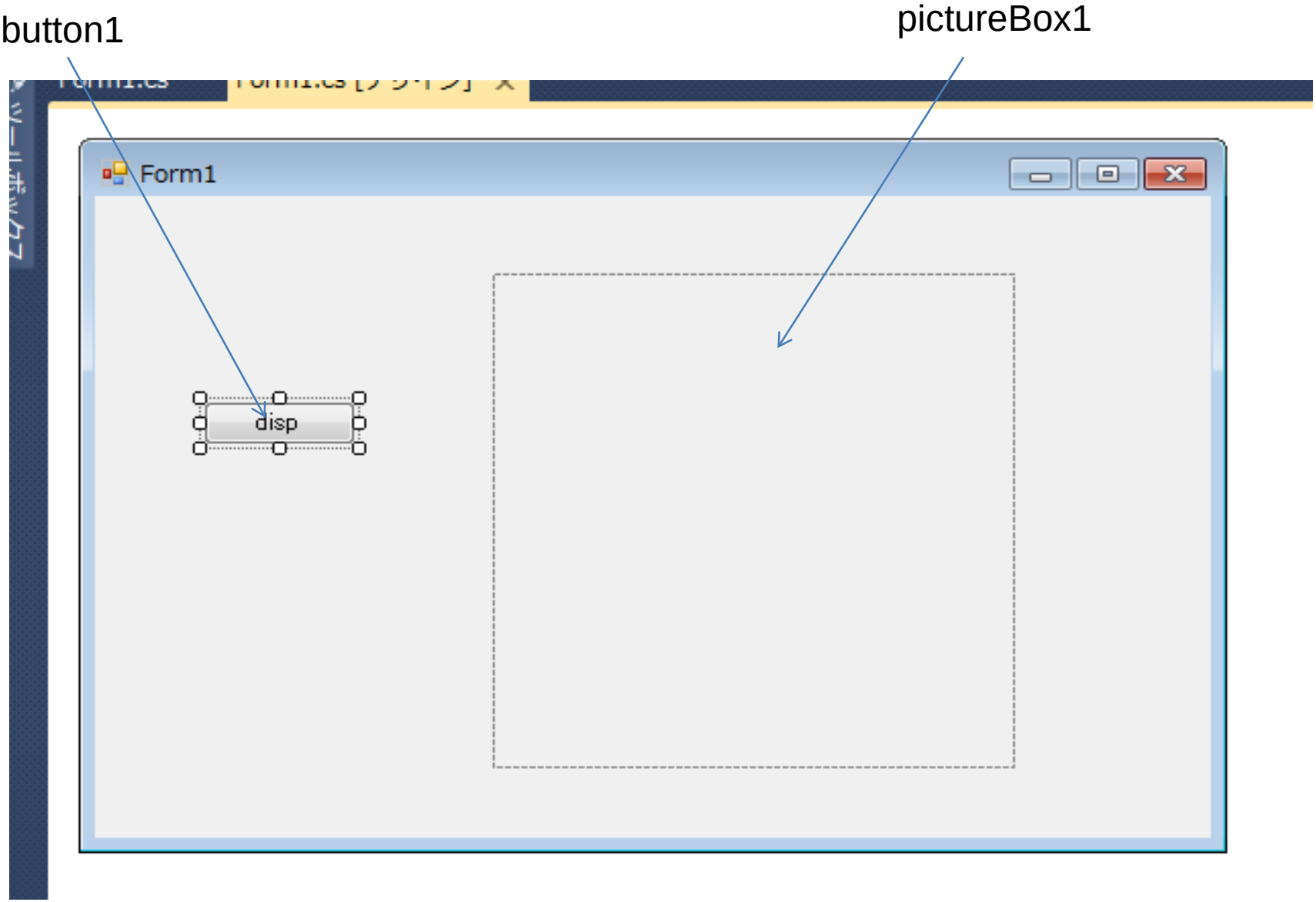


フォーム作り

- pictureBoxを右クリック→プロパティより”Size”の値を”256,256”に変更する。

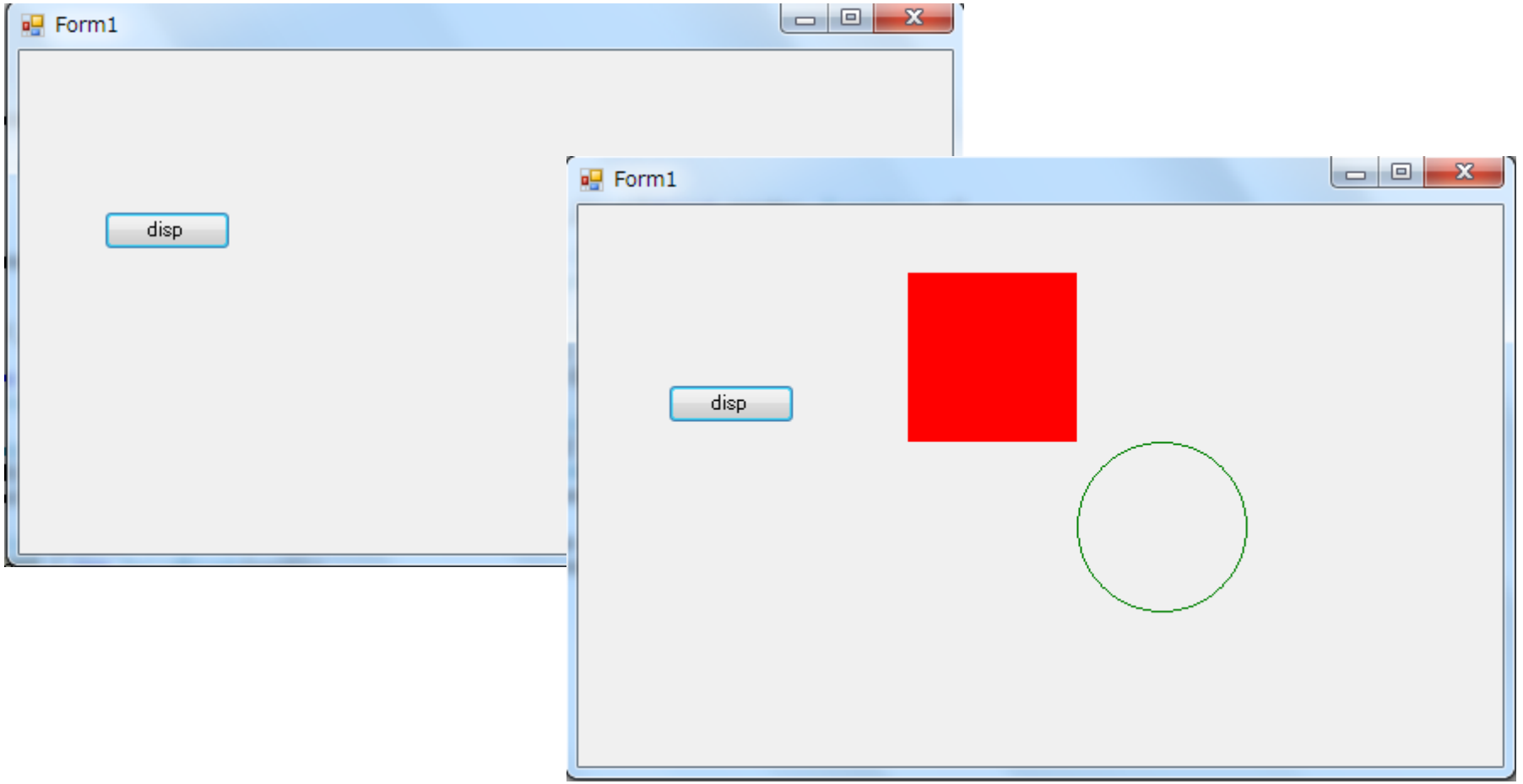


フォーム作り



今回作るプログラム

- pictureBoxに図形を描画するプログラムを作る。



コード記述

- Buttonをダブルクリックすると以下の様なコードが表示される

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace test
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {

      }


   }
}
```

コード記述

- Buttonをダブルクリックし表示されたコードに以下の様に記載する。

```
private void button1_Click(object sender, EventArgs e)
    {
        Bitmap bmp = new Bitmap(256, 256);
        Graphics gr = Graphics.FromImage(bmp);

        Color col1 = Color.Red;
        SolidBrush brush1 = new SolidBrush(col1);
        gr.FillRectangle(brush1, 0, 0, 100, 100);

        Color col2 = Color.Green;
        SolidBrush brush2 = new SolidBrush(col2);
        Pen pen2 = new Pen(brush2);
        gr.DrawEllipse(pen2, 100, 100, 100, 100);

        pictureBox1.Image = bmp;
        pictureBox1.Refresh();

    }
```

コード解説

```
Bitmap bmp = new Bitmap(256, 256);
```

- ここで、Bitmapクラスのbmpという変数を宣言する
- クラスは配列型と同様newを用いて宣言を行うことができる(配列型は厳密にはクラスであるため)

コード解説　クラス

- Aというクラスのaという変数を宣言する場合以下の様に宣言を行う

A a =new A();

- 右辺の()にはAの初期設定が入り、この関数をコンストラクタと呼ぶ
- 例えば、Bitmapクラスでは画像の大きさを決める(今回は256×256に設定している)

コード解説　Bitmapクラス

- Bitmapクラスは主に画像の表示や描画を行うクラスである。

- しかし、pixel単位での描画となるため図形や文字の描画は難しい

- 次に説明するGraphicクラスを用いると図形や文字の描画を行うことができる。

コード解説

```
Graphics gr =Graphics.FromImage(bmp);
```

- この文では、前のコードで作ったBitmapクラスのbmpからGraphicクラスのgrを作成している。
- この様に、newで作成せずにクラス自身の関数でクラスが宣言される事もある。
- 以降はこのgrを用いて描画を用いて描画を行う。

コード解説　Graphicsクラス

```
Graphics gr =Graphics.FromImage(bmp);
```

- Bitmapクラス上に図形や文字を描画できるクラス

- この方法で宣言された場合、Bitmapクラスであるbmp上に描画されることになる

コード解説　Color構造体

```
Color col1 = Color.Red ;
```

- Color構造体のCol1を宣言する。
- ここでは、クラスと構造体はほぼ同等のものと考えて良い。
- 基本的には、以下の構文で宣言することができる

```
Color col= Color.色名 ;
```

コード解説　SolidBrushクラス

- SolidBrushクラスはGraphicクラス上に絵を描くために使うブラシのクラスである。
- 他にもBrushクラスはあるが、主に色の設定に用いる。
- 基本的には以下の様に宣言できる

```
Color   col =Color.色名;
SolidBrush  brush = new SolidBrush(col);
```

これで、色名のブラシを作ることができる。

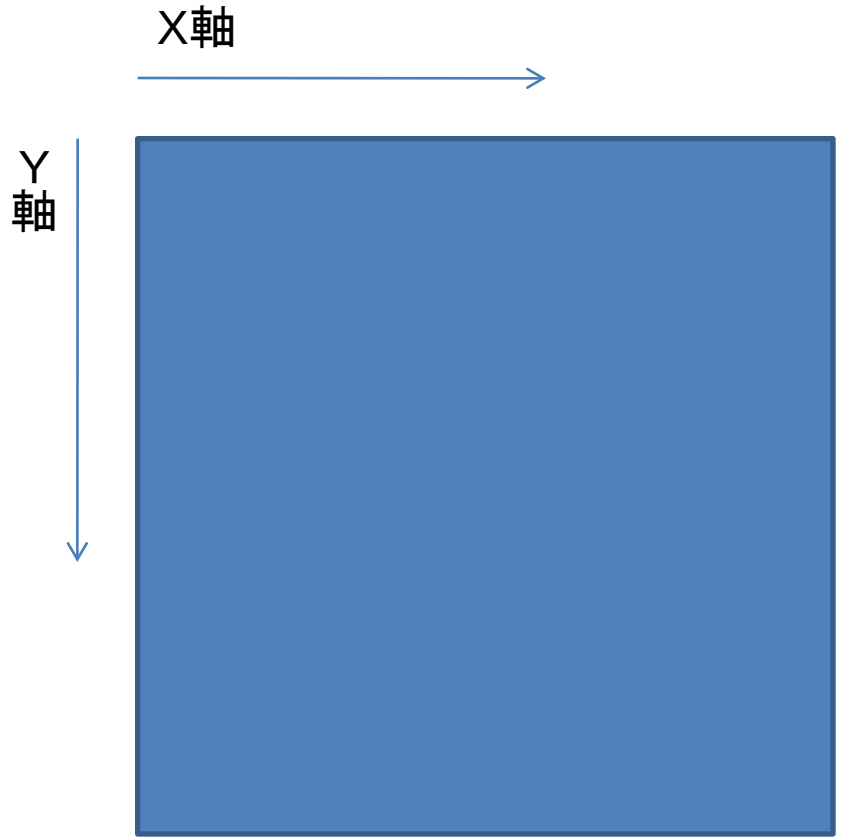
コード解説　FillRectangleメソッド

- グラフィッククラスに塗りつぶされた四角形を描くためのメソッド。
- メソッドとは、クラス内の関数の事でありこのメソッドはGraphicクラスに属している。
- つまり、「Graphicクラスgr上に塗りつぶされた四角形を描画する」という意味なる。

プログラムにおける座標の考え方

プログラム上では原点を左上として考える

(中学・高校での座標の取り方とは違うので注意)



コード解説　FillRectangleメソッド

```
gr.FillRectangle(brush1, 0, 0,  100, 100);
```

このコードは、左上の座標を(0，0)とし、幅100、高さ100、brush1の色の四角形をGraphicクラスgrに描画するという意味になる。

コード解説

```
Color col2 = Color.Green ;
SolidBrush brush2 = new SolidBrush(col2);
```

緑色のbrush2を宣言する。

コード解説　Penクラス

```
Pen pen2 = new Pen(brush2);
```

次の描画メソッドでは線を引くためそのための
Penクラスを宣言する。

ここでは、brush2の色(緑色)のPen2を作った
事になる。

コード解説　DrawEllipseメソッド

- 基本的には、FillRectangleと同じような動作をし、「Graphicクラスに中を塗りつぶさない円を書きなさい」という意味になる。

```
gr.DrawEllipse(pen2, 100, 100, 100, 100);
```

- このコードは、緑色のペンで左上の原点(100、100)で幅100、高さ100の正方形に接する円を書きなさいという意味になる。

コード解説

```
pictureBox1.Image = bmp;
pictureBox1.Refresh();
```

- pictureBox1はフォーム作りの際に作ったpictureBoxの事である。
- この2文はここまでGraphicクラスを経由して作ったBitmapクラスbmpをpictureBox1のImageとして設定し、pictureBox1を再描画するという意味になる。

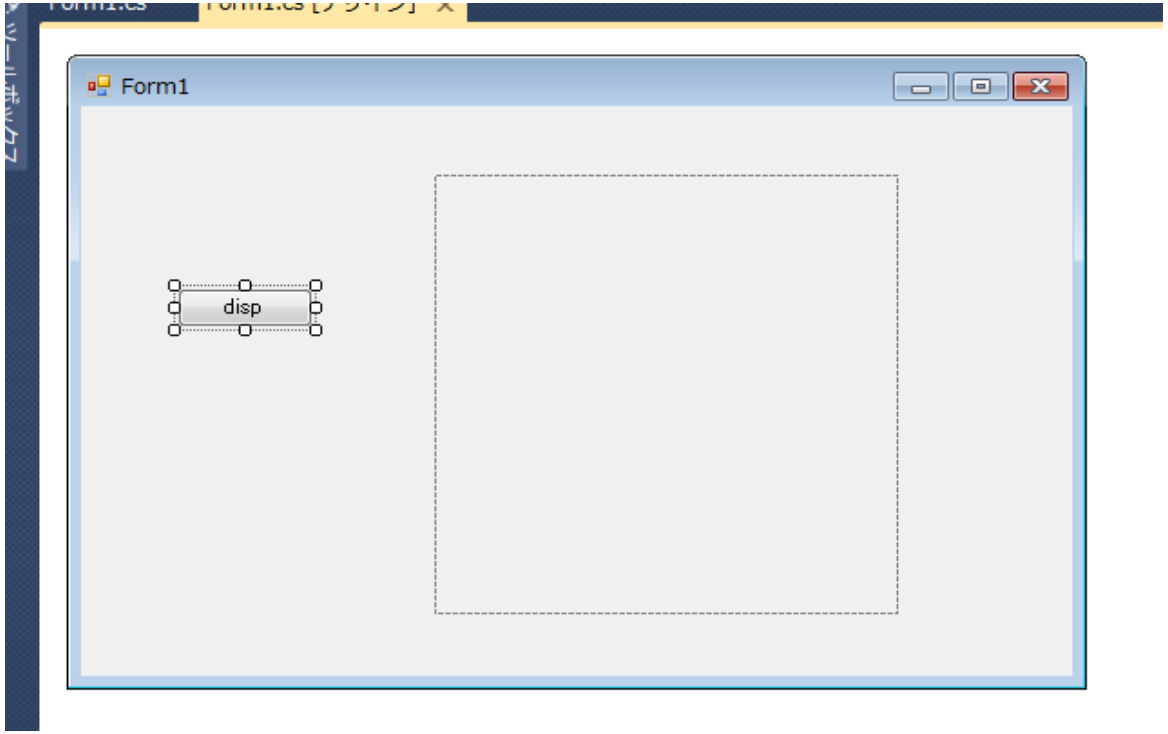
Bitmapクラスを用いた描画法

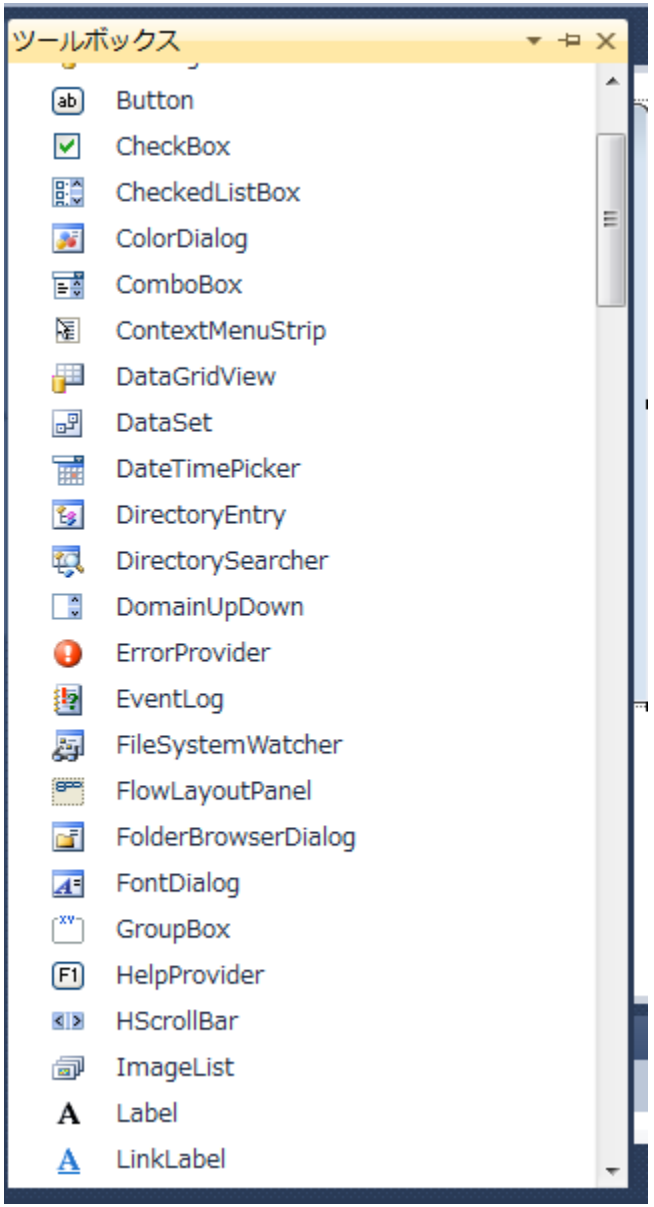
- Graphicクラスを用いない描画法を紹介する。

- バイナリ画像を扱う場合にはこちらの方法を多く用いる。

フォーム作り

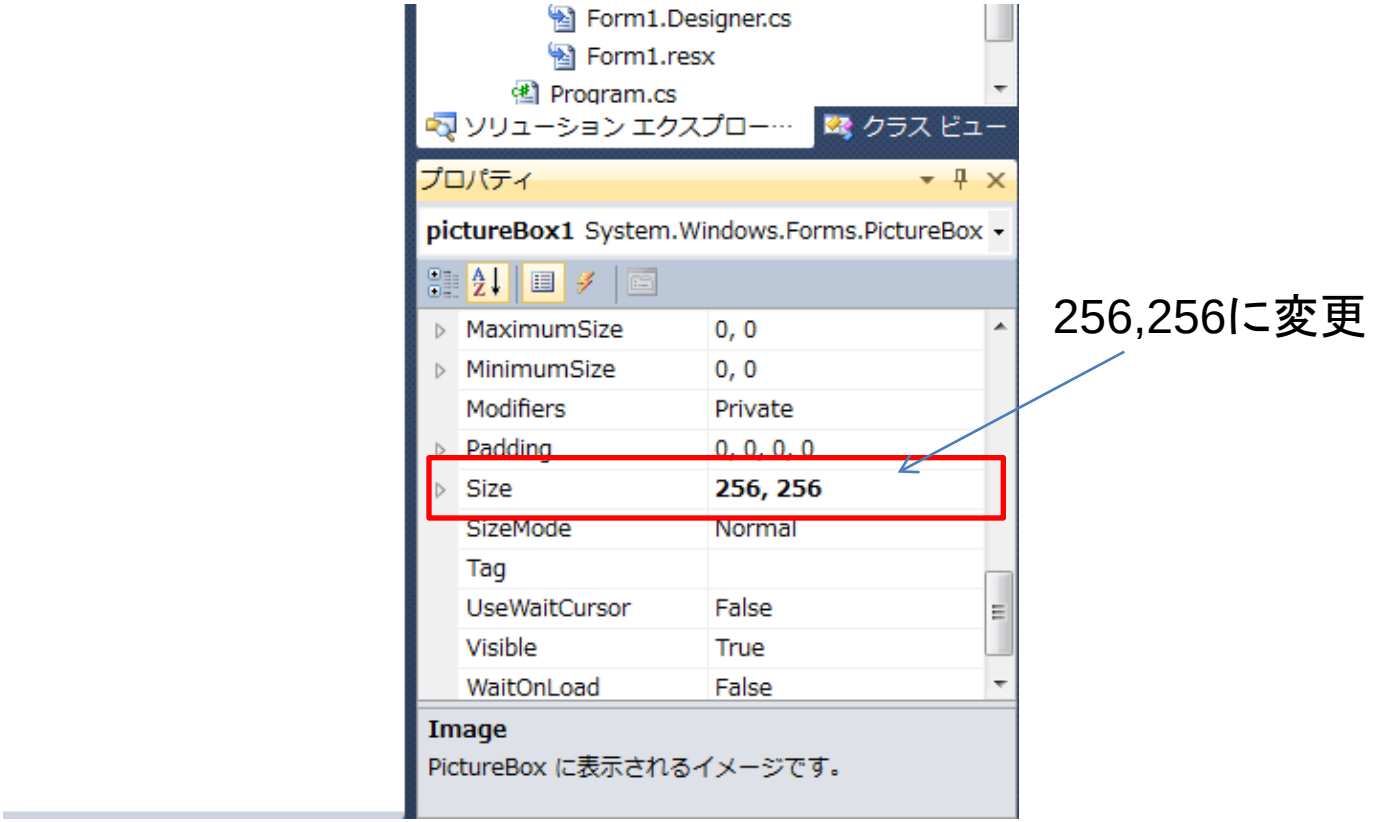
- まず、label、pictureBoxを配置する。
- ツールボックスより左クリックで選択する。
- そのまま、フォーム上を左クリックすると設置することができる。



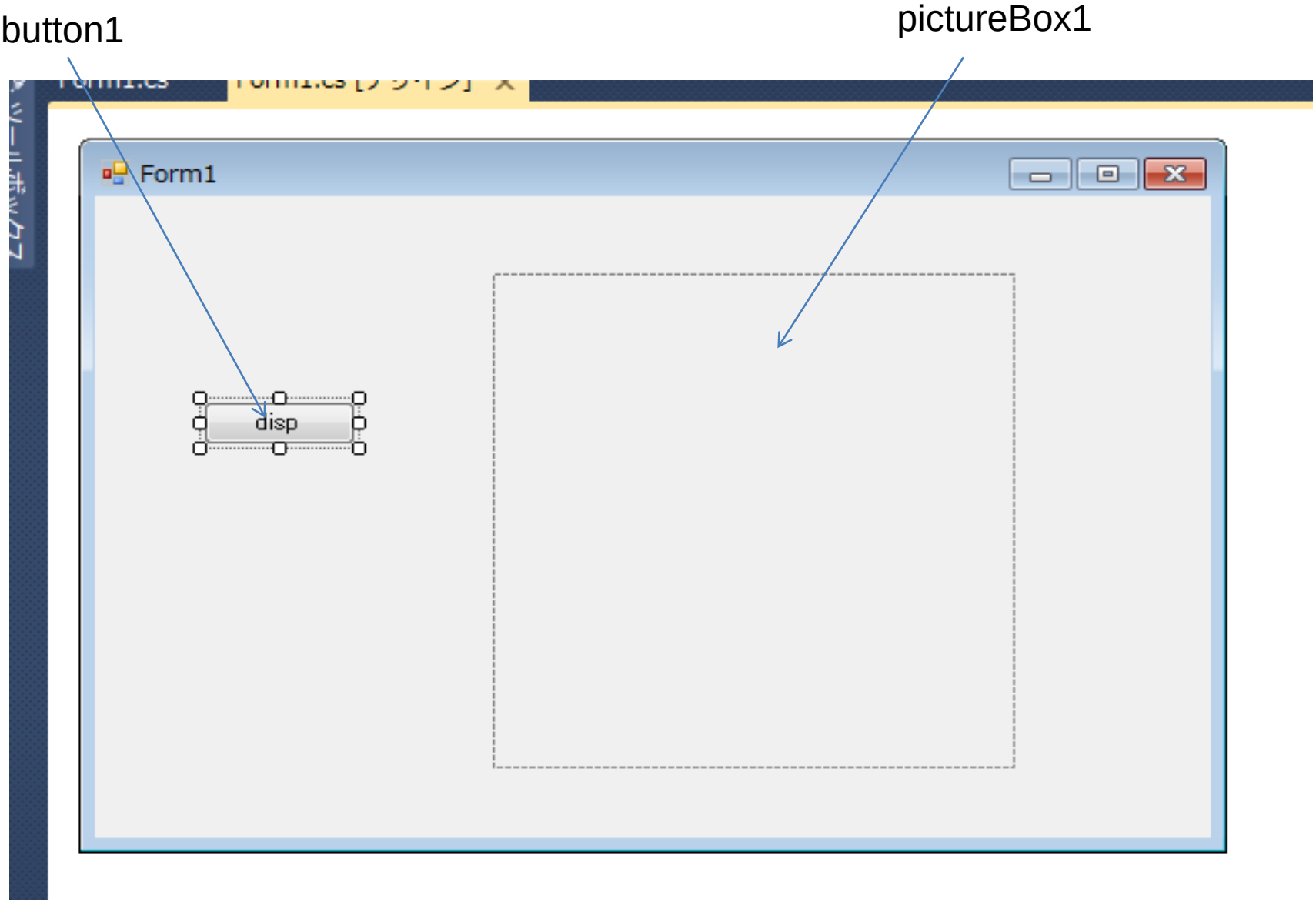


フォーム作り

- pictureBoxを右クリック→プロパティより”Size”の値を”256,256”に変更する。

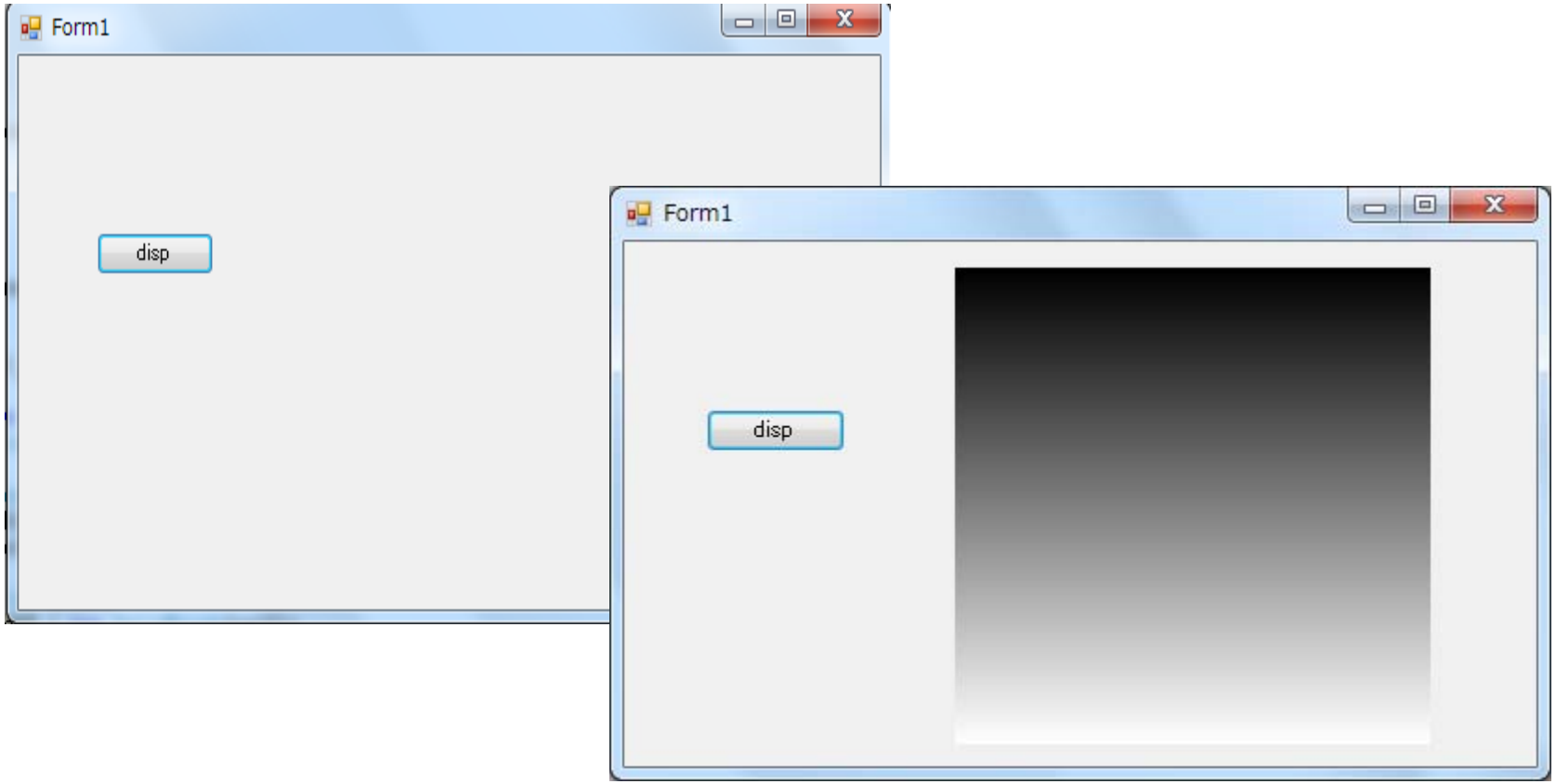


フォーム作り



今回作るプログラム

- 以下の様なグラデーションを描画する



コード記述

- Buttonをダブルクリックすると以下の様なコードが表示される

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace test
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {

      }


   }
}
```

コード記述

- Buttonをダブルクリックし表示されたコードに以下の様に記載する。

```
private void button1_Click(object sender, EventArgs e)
{
   int[,] img =new int[256,256];

   for (int j = 0; j < 256; j++)
   {
      for (int i = 0; i < 256; i++)
      {
         img[i, j] = j;
      }
   }

   Bitmap bmp = new Bitmap(256, 256);


 for (int j = 0; j < 256; j++)
 {
    for (int i = 0; i < 256; i++)
    {
       int count = img[i, j];
       Color col =Color.FromArgb(count,count,count);
       bmp.SetPixel(i, j, col);
    }
 }

   pictureBox1.Image = bmp;
   pictureBox1.Refresh();

}
```

コード解説

int[,] img =new int[256,256];

- imgという256×256の2次元配列を宣言する。
- 2次元配列とは、2次元のマス目上にデータを配置するイメージである。以下に、3×3の配列のイメージを記載する。

[0, 0]	[1, 0]	[2, 0]
[0, 1]	[1, 1]	[2, 1]
[0, 2]	[1, 2]	[2, 2]

コード解説

```
for [int j = 0; j < 256; j++]
        {
            for [int i = 0; i < 256; i++]
            {
                img[i, j] = j;
            }
        }
```

- ここでグラデーションのもとになる配列を作る。
- このコードでは0行目には0、1行目には1、2行目には2・・・・・・・・255行目には255という配列になる

コード解説

- ３×3の配列ではこんなイメージ

[0，0]　値　0	[1，0]　値　0	[2，0]　値　0
[0，1]　値　1	[1，1]　値　1	[2，1]　値　1
[0，2]　値　2	[1，2]　値　2	[2，2]　値　2

コード解説

```
Bitmap bmp = new Bitmap（256, 256）;
```

- 幅256、高さ256のBitmapクラスbmpを宣言する。
- ここは解説済みである。

コード解説

```
for (int j = 0; j < 256; j++)
        {
           for (int i = 0; i < 256; i++)
           {
              int count = img[i, j];
              Color col =Color.FromArgb(count,count,count);
              bmp.SetPixel(i, j, col);
           }
        }
```

- Color.FromArgb（赤、緑、青）を用いるとは赤、緑、青をそれぞれ0～255までの整数で記載し、任意の色を作ることができる。
- 上記の様に、赤緑青を同じ値にすると256諧調の白黒画像を作ることができる

コード解説　SetPixelメソッド

bmp.SetPixel(i, j, col);

- このメソッドでは、Bitmapの横i番目、縦j番目にcolの画素を設置できる。
- ここでは、bmpに対し行っている。

コード解説

- 3×3の配列ではこんなイメージ

[0, 0]	値 0	[1, 0]	値 0	[2, 0]	値 0
[0, 1]	値 1	[1, 1]	値 1	[2, 1]	値 1
[0, 2]	値 2	[1, 2]	値 2	[2, 2]	値 2

- 値が大きいほど白くなり、インデックスにより配置される場所が決まる
- このように、Bitmap上にグラデーションを描画することができる。

コード解説

```
pictureBox1.Image = bmp;
pictureBox1.Refresh();
```

- 前にも説明したように、このコードを記述することにより作ったBitmap画像を描画する事が出来る。

まとめ

- 今回は2つの方法による描画を行った。
- Graphic系のメソッドは数多くあるため興味がある場合は調べて見てください。
- SetPixelメソッドを用いた描画は画像処理での使用頻度が高いためしっかり確認して下さい。