1. Open a new Project: CLR/Windows Form Application

2. We will be using Drawing objects. The form should contain one pictureBox and two buttons. We will use one button to draw objects, and the other button to clear the pictureBox. Open the Toolbox (using View/ToolBox from the bar menu), if it is not open already. Drag a pictureBox and two buttons onto your form.
3. Select each button and look at the properties of that button. If you don’t see the Properties window, right click on the button to open it. Change the Text property of the first button to **Draw!** and the second button to **Clear!**
4. Double click next to one of the buttons on the form. This should bring you to the form1.h file, with an empty function named Form1_Load. Immediately above the function, and right after the line #pragma endregion, place the following line of code:

```cpp
Drawing::Graphics^ g;
```

`g` is a Graphics object that is in the `Drawing` class (notice the scope resolution operator ::)

The `^` is called a “handle” and it is used in C++/CLR. It is similar to the * (pointer declaration) that we covered in class. The difference between them is beyond the scope of this project, except to say that the programmer must allocate objects declared with ^ using `gcnew`, but they are garbage collected automatically.
4. For the body of the function `Form1_Load`, place the following line of code.

```csharp
    g = pictureBox1.CreateGraphics();
```

This line of code creates an instance of the `Graphics` object. The `Graphics` object represents a drawing surface, and is the object that is used to create graphical images (http://msdn.microsoft.com/en-us/library/5y289054.aspx)
5. You can compile and run your code, if you wish. Press CTL-F5
6. You can click on the Form1.h (design) tab to get back to the design window.

7. Double click on the Draw! button to get to the event handler button1_Click (this is a function or method that is called each time the button is clicked) called. Between the curly braces of this function (i.e., in the body of the function), type the following. These are just examples, you can use other drawing objects.

```csharp
// instances of the Pen class
// A pen is used to draw lines, curves, and to outline shapes (think about these like the c++ line)
// int * p = new int;
// the Color can be set when the object is created
// think constructors!

Pen^ redPen = gcnew Pen(Color::Red);
Pen^ blackPen = gcnew Pen(Color::Black);

// instance of Brush class

Brush^ greenBrush = gcnew Drawing::SolidBrush(Color::Green);

//declare a rectangle called circleRect. The first two //parameters correspond to the x and y position on the //form of the top left corner of the rectangle, and the
//next two parameters are the width and height of the rectangle. You can think of the top left corner of the form as position (0,0), and x increases to the right and y increases downward
Rectangle circleRect(45,30,100,100);

//call the function FillEllipse using the graphics object //g. The two parameters are the brush and the rectangle. //This function fills a circle. Try other function such as //FillRectangle or FillPolygon

g->FillEllipse(greenBrush,circleRect);

//set Width of the Pen
redPen->Width = 5.0F;
blackPen->Width = 3.0F;

//declare Point objects, parameters are x,y coordinates
Point startPoint = Point(120, 30);
Point endPoint = Point(120, 80);

//call DrawLine method (function) sending three parameters: //a Pen, and two points. This draws a line on the graphics //object, with the pen from the first point to the second //point
g->DrawLine( redPen, startPoint, endPoint);
g->DrawLine( redPen, startPoint, endPoint);

//call other methods. DrawArc and DrawPie take a Pen, a //Rectangle and a start point and number of degrees
g->DrawArc(redPen, circleRect, 25.0, 180.0);
g->DrawPie(blackPen, circleRect, 180.0, 25.0);
8. Compile and run your program. Click the Draw button. The shapes and lines are drawn. Probably you can use the methods and objects to draw something a bit nicer. Try this code in the draw function:

```cpp
Pen^ redPen = gcnew Pen(Color::Red);
redPen->Width = 5.0F;
Point startPoint = Point(120, 30);
Point endPoint = Point(120, 80);
g->DrawLine( redPen, startPoint, endPoint);

startPoint.X = 150; //changes X coordinate
endPoint.X = 150;
g->DrawLine( redPen, startPoint, endPoint);
startPoint.X = 170;
endPoint.X = 170;
g->DrawLine( redPen, startPoint, endPoint);
startPoint.X = 190;
endPoint.X = 190;
g->DrawLine( redPen, startPoint, endPoint);
startPoint.X = 190;
endPoint.X = 190;
startPoint.Y = 85;
endPoint.Y = 90;
g->DrawLine( redPen, startPoint, endPoint);
```
```csharp
startPoint.X = 120;
endPoint.X = 150;
startPoint.Y = 55;
endPoint.Y = 55;
g->DrawLine( redPen, startPoint, endPoint);
```

9. Double Click on the clear button and add one line to the body of the event handler as seen below in the screen shot.

References

2. http://www.youtube.com/watch?v=klty_vu20qY