Introduction

I wrote the original code for my Sudoku program using Visual Basic on Microsoft's Visual Studio IDE. This restricts the program to Windows. To allow the code to execute successfully on Windows and Linux, I decided to use the portability of the C++ language.

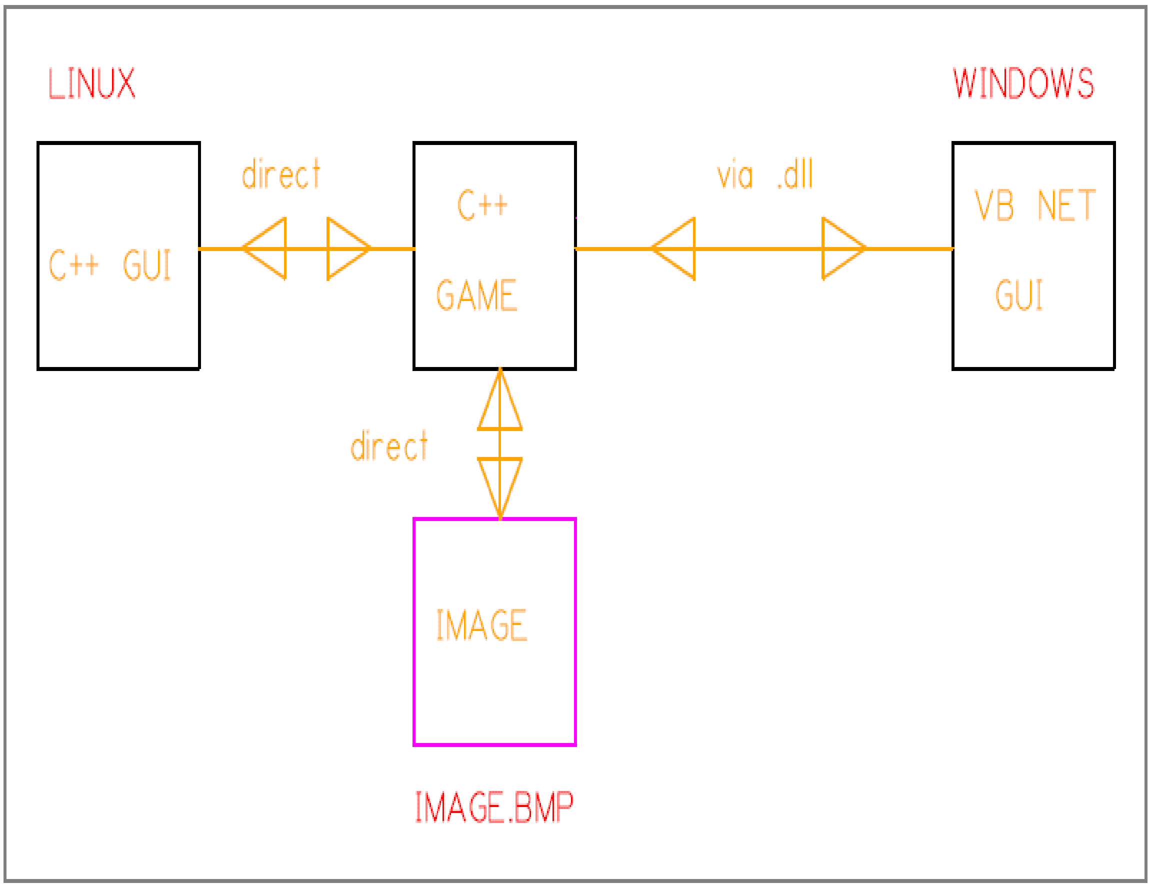
I documented this in a previous article:

http://www.codeproject.com/KB/miscctrl/cross_platform.aspx

My approach received much criticism as the “same code” could not be compiled directly into a platform application.

I abstracted the program to a series of functions which can be interfaced to the native GUI toolkits.

The GUI interface needs only some buttons, an image box and mouse position. It is used only to call the functions in the core C++ program.



In Windows, I still use Visual Basic IDE to provide the GUI. The functions are provided by compiling the C++ core to a managed assembly DLL.

In Linux, I use Glade to develop the GUI and associated code. The functions are added to the GUI code directly.

The image of the Sudoku uses an uncompressed bitmap file in Windows BMP file format. This can be displayed in Windows and Linux. It can also be directly worked on to produce the various images necessary for the program.

I still standby this method of generating cross-platform applications.

It is especially useful when we have:

1. Most of the operational code able to be abstracted to a block of C++.

2. Only a simple GUI is needed on the platform.

This does require individual compilation for each platform that it supports.

Another approach is to use code that can be directly run on any platform without special preparation. This could be software written in an interpreted language for which run-time packages are needed.

I have been learning java so I decided to rewrite my sudoku program and make it truly platform independent.

Lets hope it will be “Write Once, Run Anywhere”, not “Write Once, Debug Everywhere”.

We can use Eclipse or NetBeans IDE. Each has is own GUI toolkit.

I do use Eclipse but I decided to use Java Foundation Classes (JFC) and Swing.

The code can be written and hacked with a text editor using javac to compile

and java to run the application.

Game Code

The operational code needed to run the game is in:

```
public class Smethods
{
public static byte select(byte[][] sudoku, byte number, byte position, byte step)
      {
      if((sudoku[position*9 + number][step] == 0) || (sudoku[position*9 + number]
[step] > 9))
            return step;//end of number not possible or is selected

    step += 1; // we can select so write this step to the sudoku array
      int count = 0;
      for(count = 0; count < 729; count++)
            sudoku[count][step] = sudoku[count][step - 1];
//copy existing to next step
      for(count = 0; count < 9; count++)
            sudoku[position*9 + count][step] = 0;   //Can't select any in box

      byte row =   (byte) (position/9);
      for(count = 0; count < 9; count++)
            sudoku[row * 81 + count * 9 + number][step] = 0; //horizontal row

      byte column =   (byte) (position%9);
      for(count = 0; count < 9; count++)
            sudoku[column * 9 + count * 81 + number][step] = 0;  //vertical row

      int brow =  (position/27)*243; //row block 0f 3
      column = (byte) (((position%9)/3)*27);  //Column block of 3
      byte incount;
      for(incount = 0; incount < 3; incount++)
      {
      for(count = 0; count < 3; count++)
            sudoku[brow + column + count * 9 + incount * 81 + number ][step] =
0;  //box of 3 x 3
      }//end of 3 x 3 box
      //we have selected one number
      sudoku[position*9 + number][step] = (byte) (number + 11); //selected now 11
to 19
      return step;
      }//end of select a number

}//end of class
```

I have only shown one method in the Smethods class.
We also have:

```
public static void start(byte[][] sudoku)
public static void trysudoku(byte[][] sudoku, byte startstep)
```

This provides all the calculations necessary to generate and solve sudoku games.

Display Window

This is created by:

```
public class MySudoku
{
        public static byte[][] sudoku = new byte[729][82];  //global array for
sudoku solution
        public static byte step = 0; //global variable for solution step

        private static final int WindowWidth = 777; //its 777 pixels wide
        private static final int WindowHeight = 636; //its 636 pixels high

    public static void ShowGUI()
       {
          Smethods.start(sudoku); //start array at step 0 has no numbers selected

                final byte border = 14;  //border for display
                JFrame f = new JFrame("MySudoku");
              f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
              BufferedImage image = null;
                  try {
                              image = ImageIO.read(new File("sudoku.png"));
                       } catch (IOException e) {
                       }//end of try/catch
              f.setResizable(false);  //not to be resized
              f.setIconImage(image);
                f.setSize(WindowWidth, WindowHeight);
//size fixed by size of display and borders
                f.setLocation(0, 0); //start top left
                f.setLayout(new BorderLayout());
//north south east west and centre

                f.add(new SPanel(new Dimension(WindowWidth,border)),
BorderLayout.NORTH);
```

```
                 f.add(new SPanel(new Dimension(WindowWidth,border)), 
BorderLayout.SOUTH);
                 f.add(new SPanel(new Dimension(border,WindowHeight)), 
BorderLayout.EAST);
                 f.add(new SPanel(new Dimension(0,WindowHeight)),   BorderLayout.
WEST);  //set the borders

              DisplayPanel dp =new  DisplayPanel();
              dp.setBackground(Color.BLACK);  //set the background of the sudoku 
display black
              f.add(dp, BorderLayout.CENTER);  //add the sudoku display panel

              f.setVisible(true);
        }//end of show gui method

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
             ShowGUI();
            }
        });  //end of run()
    }//end of main

}//end of MySudoku class
```

Display Panel

This is where all the hard work is done to create the GUI.
It positions and monitors the push buttons.

It paints the sudoku game display:

```
public class DisplayPanel extends JPanel implements ActionListener
//create display panel
{
      private static final long serialVersionUID = 1L;
      private int DisplayWidth = 557; //sudoku display its 557 pixels wide
    private int DisplayHeight = 580; //sudoku display its 580 pixels high
    private int ButtonsWidth = 200; //button panel its 200 pixels wide
    private final Color LB = new Color(0xAD,0xD8, 0xE6);  //Light Blue
    private final Color DB = new Color(0x1E,0x90, 0xFF);  //dodger blue
    private final Color P = new Color(0x80,0, 0x80);  //purple blank number

    public DisplayPanel()   //construct the sudoku display panel
    {
        addMouseListener(new MouseAdapter() 
//we listen for mouse clicks on this panel
        {
            public void mousePressed(MouseEvent e)
            {
                selectNumber(e.getX(),e.getY()); 
//the called method on mouse click
            }//end of mouse select
        });//end of mouse listener
        this.setLayout(new BorderLayout());

        JPanel pb = new JPanel();   //create the button panel
        pb.setPreferredSize(new Dimension(ButtonsWidth,DisplayHeight));
        pb.setBackground(LB);

        FlowLayout FL = new FlowLayout();
        FL.setVgap(55);
        FL.setHgap(100);  //set the flow layout to give  symmetric display
        pb.setLayout(FL);
        SButton EYS = new SButton("Enter Your Sudoku", "EYS");
        EYS.addActionListener(this);
        pb.add(EYS);
        SButton SHS = new SButton("Start Hard Sudoku", "SHS");
        SHS.addActionListener(this);
        pb.add(SHS);
        SButton SMS = new SButton("Start Medium Sudoku", "SMS");
        SMS.addActionListener(this);
        pb.add(SMS);
        SButton SES = new SButton("Start Easy Sudoku", "SES");
        SES.addActionListener(this);
        pb.add(SES);
        SButton GBS = new SButton("Go Back One Step", "GBS");
        GBS.addActionListener(this);
        pb.add(GBS);
        SButton STS = new SButton("Solve This Sudoku", "STS");
        STS.addActionListener(this);
        pb.add(STS);
        this.add(pb,BorderLayout.WEST);
//add the push button panel to the display panel
    }//end of constructor
}//end of my display panel class
```

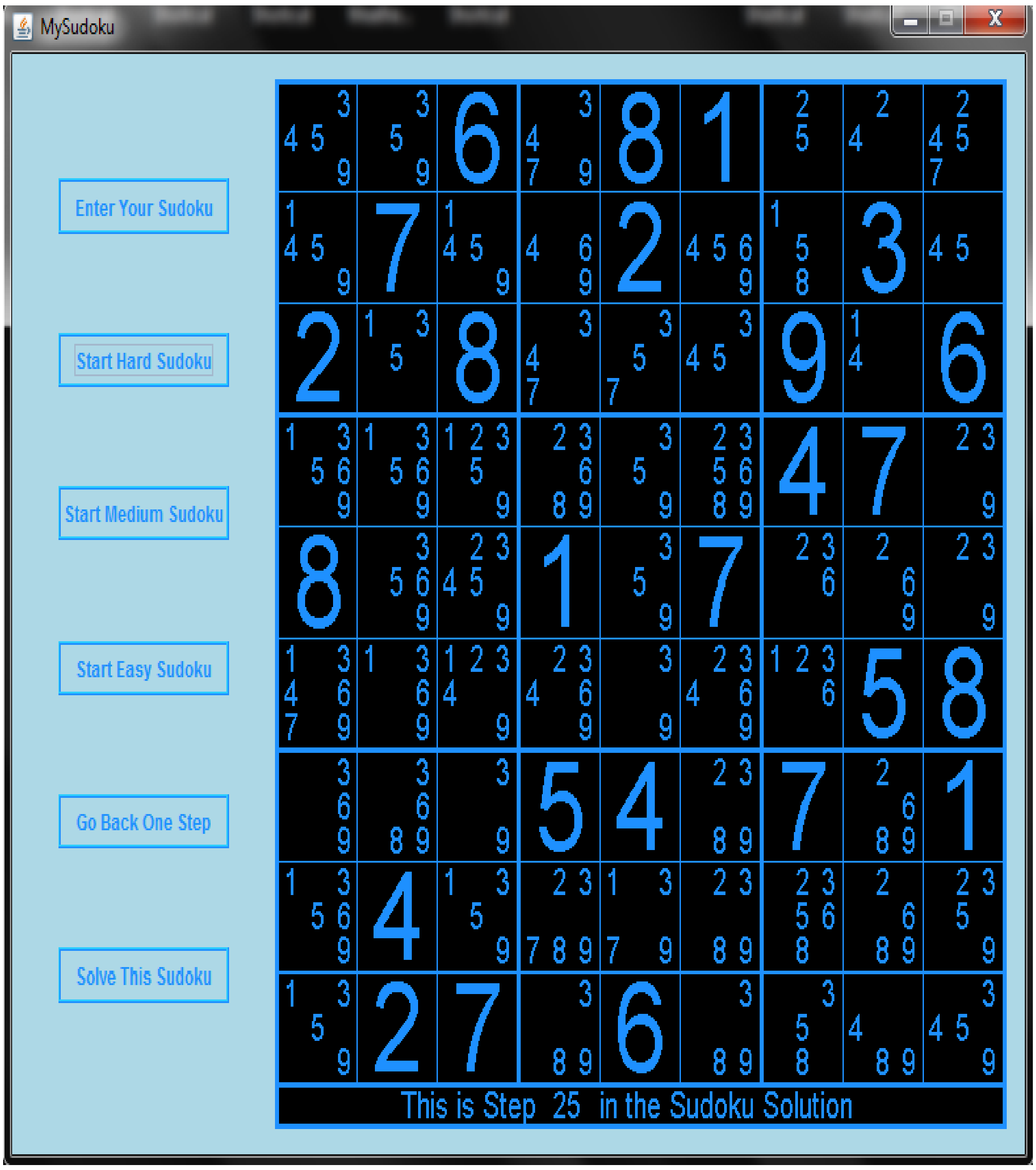
It have only shown the constructor we also have:

```
private void selectNumber(int x, int y)   //called method on mouse click
public Dimension getPreferredSize()  //set the preferred size of display panel
protected void paintComponent(Graphics g)
//called whenever the display panel needs painting
public void actionPerformed(ActionEvent e)  //call method for push button selected
```

We also have two other classes used to create push buttons and panels.

```
class SButton extends JButton   //create a JButton with some fixed properties
public class SPanel extends Panel   //create border panels for the display
```

We then get the following window:



This is Step 25 in the Sudoku Solution

Conclusions

It have only been able to test it on windows XP, windows 7 and vista.
These machines had the correct java run time installed.

If someone could spare the time it would be nice to know if it worked on Linux or Mac.