

Background color = Control

using System.Data;

using System.Drawing;

using System.Runtime.InteropServices;

using System.Windows.Forms;

namespace ReAgeEligibilty

{

public partial class MainForm : Form

{

public const int WM\_NCLBUTTONDOWN = 0xA1;

public const int HT\_CAPTION = 0x2;

public int failed = 0;

private Color BackGroundAlternateColor = Color.LightBlue;

[DllImportAttribute("user32.dll")]

public static extern int SendMessage(System.IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImportAttribute("user32.dll")]

public static extern bool ReleaseCapture();

[DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]

private static extern System.IntPtr CreateRoundRectRgn

(

int nLeftRect, // x-coordinate of upper-left corner

int nTopRect, // y-coordinate of upper-left corner

int nRightRect, // x-coordinate of lower-right corner

int nBottomRect, // y-coordinate of lower-right corner

int nWidthEllipse, // height of ellipse

int nHeightEllipse // width of ellipse

);

public MainForm()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.None;

Region = System.Drawing.Region.FromHrgn(CreateRoundRectRgn(0, 0, Width, Height, 20, 20));

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void MainForm\_Load(object sender, System.EventArgs e)

{

label2.Text = "All Are Eligible";

Grid1.RowHeadersVisible = false;

// HEADER

// INCREASE HEADER HEIGHT

Grid1.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.EnableResizing;

Grid1.ColumnHeadersHeight = 40;

// BACKGROUND AND FOREGROUND COLORS

Grid1.EnableHeadersVisualStyles = false;

// Bold, Font, Size

Grid1.ColumnHeadersDefaultCellStyle.Font = new Font("Tahoma", 9.75F, FontStyle.Bold);

// TAKE AWAY GRID LINES ON HEADER

Grid1.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.None;

// ALLOW HORIZONTAL LINES ONLY

Grid1.CellBorderStyle = DataGridViewCellBorderStyle.SingleHorizontal;

// LOAD

GetData();

}

#region DEFINE STRUCTURE

private DataTable DefGridHeader()

{

DataTable Data1 = new DataTable();

Data1.Columns.Add("", typeof(string));

Data1.Columns.Add("Test", typeof(string));

Data1.Columns.Add("Result", typeof(string));

return Data1;

}

#endregion

#region LoadData

private void GetData()

{

DataTable Datas = DefGridHeader();

Datas.Rows.Add("", "", "");

Datas.Rows.Add("", "Test Account 1", "Eligible");

Datas.Rows.Add("", "Test Account 2", "Non Eligible");

Datas.Rows.Add("", "Test Account 3", "Eligible");

Datas.Rows.Add("", "Test Account 4", "Eligible");

Datas.Rows.Add("", "Test Account 5", "Eligible");

Datas.Rows.Add("", "Test Account 6", "Eligible");

Datas.Rows.Add("", "Test Account 7", "Eligible");

Datas.Rows.Add("", "Test Account 8", "Non Eligible");

Datas.Rows.Add("", "Test Account 9", "Non Eligible");

Datas.Rows.Add("", "Test Account 10", "Eligible");

Datas.Rows.Add("", "Test Account 11", "Eligible");

Datas.Rows.Add("", "Test Account 12", "Eligible");

Grid1.DataSource = Datas;

// DRAW FOR FIRST ROW LIKE A BLACK LINE

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if (Grid1.Rows.Count > 0)

{

Grid1.Rows[0].DefaultCellStyle.BackColor = Color.Black;

Grid1.Rows[0].Height = 3; // LOOK LIKE A LINE\*\*\*\*\*\*

}

// Grid1.ColumnHeadersHeight = Grid1.Rows[0].Height \* 3;

// ALTERNATE ROW BACKGROUND COLOR

for (int i = 1; i < Grid1.Rows.Count; i++)

{

if (i % 2 != 0) Grid1.Rows[i].DefaultCellStyle.BackColor = BackGroundAlternateColor;

}

// SET REAL HEIGH OF GRID

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Grid1.Height = (Grid1.Rows.Count \* 22) + 30;

Grid1.Columns[0].Width = 0;

Grid1.Columns[1].Width = 270;

Grid1.Columns[2].Width = 70;

}

#endregion

#region rectangles

static void DrawRoundedRectangle(Graphics g, Pen p, int x, int y, int w, int h, int rx, int ry)

{

System.Drawing.Drawing2D.GraphicsPath path = new System.Drawing.Drawing2D.GraphicsPath();

path.AddArc(x, y, rx + rx, ry + ry, 180, 90);

path.AddLine(x + rx, y, x + w - rx, y);

path.AddArc(x + w - 2 \* rx, y, 2 \* rx, 2 \* ry, 270, 90);

path.AddLine(x + w, y + ry, x + w, y + h - ry);

path.AddArc(x + w - 2 \* rx, y + h - 2 \* ry, rx + rx, ry + ry, 0, 91);

path.AddLine(x + rx, y + h, x + w - rx, y + h);

path.AddArc(x, y + h - 2 \* ry, 2 \* rx, 2 \* ry, 90, 91);

path.CloseFigure();

g.DrawPath(p, path);

}

private void panel1\_Paint(object sender, PaintEventArgs e)

{

DrawRoundedRectangle(e.Graphics, new Pen(Brushes.Blue, 1), 10, 10, 460, 340, 10, 10);

}

private void panel2\_Paint(object sender, PaintEventArgs e)

{

DrawRoundedRectangle(e.Graphics, new Pen(Brushes.Blue, 1), 10, 10, 160, 30, 10, 10);

}

private void MainForm\_Paint(object sender, PaintEventArgs e)

{

DrawRoundedRectangle(e.Graphics, new Pen(Brushes.Blue, 1), 4, 4, 507, 547, 4, 4);

}

#endregion

private void button3\_Click(object sender, System.EventArgs e)

{

Application.Exit();

}

// DISABLE SORT

private void Grid1\_ColumnAdded(object sender, DataGridViewColumnEventArgs e)

{

Grid1.Columns[e.Column.Index].SortMode = DataGridViewColumnSortMode.NotSortable;

}

private void Grid1\_RowPrePaint(object sender, DataGridViewRowPrePaintEventArgs e)

{

int FILA = e.RowIndex;

string SS = Grid1.Rows[e.RowIndex].Cells[2].Value.ToString();

if (SS == "Non Eligible")

{

failed++;

if (failed % 2 == 0)

{

int ss = failed / 2;

if (ss == 1)

label2.Text = ss.ToString() + "1 Non Eligible";

else

label2.Text = ss.ToString() + " Non Eligible";

}

Grid1.Rows[e.RowIndex].Cells[2].Style = new DataGridViewCellStyle { ForeColor = Color.Red };

}

}

private void MainForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HT\_CAPTION, 0);

}

}

private void label2\_Paint(object sender, PaintEventArgs e)

{

}

private void button1\_Click(object sender, System.EventArgs e)

{

}

}

}