Problem 21

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Problem\_21

{

public partial class Form1 : Form

{

private class CActorImage

{

public int X, Y, W, H;

public List<Bitmap> imgs;

public int shrink;

public int iFrame;

}

public Form1()

{

this.WindowState = FormWindowState.Maximized;

this.Load += Form1\_Load;

this.Paint += Form1\_Paint;

this.MouseDown += Form1\_MouseDown;

tt.Interval = 100;

tt.Start();

tt.Tick += Tt\_Tick;

}

Bitmap off;

Timer tt = new Timer();

List<CActorImage> LImages = new List<CActorImage>();

int ctTick=0;

private void Tt\_Tick(object sender, EventArgs e)

{

if (ctTick % 10 == 0)

{

CreateImage();

}

for (int i = 0; i < LImages.Count; i++)

{

if (LImages[i].shrink == 1 && LImages[i].W > 0 && LImages[i].H > 0)

{

LImages[i].W -= 3;

LImages[i].H -= 3;

if (LImages[i].W <= 0 || LImages[i].H <= 0)

{

LImages.RemoveAt(i);

}

}

}

ctTick++;

DrawDubb(this.CreateGraphics());

}

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

for (int i = 0; i < LImages.Count; i++)

{

if (e.X >= LImages[i].X && e.X <= LImages[i].X + 50 && e.Y >= LImages[i].Y && e.Y <= LImages[i].Y + 50)

{

LImages[i].iFrame = 1;

LImages[i].shrink = 1;

DrawDubb(this.CreateGraphics());

break;

}

}

}

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

DrawDubb(this.CreateGraphics());

}

private void Form1\_Load(object sender, EventArgs e)

{

off = new Bitmap(this.ClientSize.Width, this.ClientSize.Height);

}

void CreateImage()

{

Random rr = new Random();

CActorImage pnn=new CActorImage();

pnn.X = rr.Next(50,this.ClientSize.Width - 50);

pnn.Y = rr.Next(50, this.ClientSize.Height - 50);

pnn.W = 50;

pnn.H = 50;

pnn.imgs = new List<Bitmap>();

for (int i = 0; i < 2; i++)

{

Bitmap im = new Bitmap("image\_" + (i + 1) + ".bmp");

pnn.imgs.Add(im);

}

pnn.iFrame = 0;

pnn.shrink = 0;

LImages.Add(pnn);

}

void DrawScene(Graphics g)

{

g.Clear(Color.Black);

for (int i = 0; i < LImages.Count; i++)

{

g.DrawImage(LImages[i].imgs[LImages[i].iFrame], LImages[i].X, LImages[i].Y, LImages[i].W, LImages[i].H);

}

}

private void DrawDubb(Graphics g)

{

Graphics g2 = Graphics.FromImage(off);

DrawScene(g2);

g.DrawImage(off, 0, 0);

}

}

}

Problem 22

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Net.Mime.MediaTypeNames;

namespace Problem\_22

{

public class CActorImage

{

public Rectangle rcSrc, rcDst;

public Bitmap img;

public int pos;

}

public partial class Form1 : Form

{

public Form1()

{

this.WindowState = FormWindowState.Maximized;

this.Load += Form1\_Load;

this.Paint += Form1\_Paint;

this.MouseDown += Form1\_MouseDown;

}

Bitmap off;

List<CActorImage> LImages = new List<CActorImage>();

int fullW, fullH;

int swapX = -1, swapY = -1;

int emptyX = -1, emptyY = -1;

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

for (int i = 0; i < LImages.Count - 1; i++)

{

if (isClick(LImages[i], e.X, e.Y))

{

if (IsAdjacent(LImages[i], emptyX, emptyY))

{

swapX = LImages[i].rcDst.X;

LImages[i].rcDst.X = emptyX;

emptyX = swapX;

swapY = LImages[i].rcDst.Y;

LImages[i].rcDst.Y = emptyY;

emptyY = swapY;

}

break;

}

}

DrawDubb(this.CreateGraphics());

if (CheckWin())

{

MessageBox.Show("YAYY YOU WON!!!");

}

}

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

DrawDubb(this.CreateGraphics());

}

private void Form1\_Load(object sender, EventArgs e)

{

off = new Bitmap(this.ClientSize.Width, this.ClientSize.Height);

CreatePuzzle();

Shuffle();

}

void CreatePuzzle()

{

int y = 0;

for (int r = 0; r < 3; r++)

{

for (int c = 0; c < 3; c++)

{

CActorImage pnn = new CActorImage();

pnn.img = new Bitmap("animals\_dog.jpg");

pnn.rcSrc = new Rectangle(c \* (pnn.img.Width / 3), y, pnn.img.Width / 3, pnn.img.Height / 3);

fullW = pnn.img.Width;

fullH = pnn.img.Height;

LImages.Add(pnn);

}

y += (LImages[0].img.Height / 3);

}

LImages[2].rcDst = new Rectangle(0 \* (LImages[0].img.Width / 3), 0, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[4].rcDst = new Rectangle(1 \* (LImages[0].img.Width / 3), 0, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[7].rcDst = new Rectangle(2 \* (LImages[0].img.Width / 3), 0, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[0].rcDst = new Rectangle(0 \* (LImages[0].img.Width / 3), LImages[0].img.Height / 3, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[6].rcDst = new Rectangle(1 \* (LImages[0].img.Width / 3), LImages[0].img.Height / 3, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[3].rcDst = new Rectangle(2 \* (LImages[0].img.Width / 3), LImages[0].img.Height / 3, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[1].rcDst = new Rectangle(0 \* (LImages[0].img.Width / 3), (LImages[0].img.Height / 3) \* 2, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[5].rcDst = new Rectangle(1 \* (LImages[0].img.Width / 3), (LImages[0].img.Height / 3) \* 2, LImages[0].img.Width / 3, LImages[0].img.Height / 3);

LImages[8].img = new Bitmap("red.png");

LImages[8].rcDst.X = 2 \* (LImages[0].img.Width / 3);

LImages[8].rcDst.Y = 2 \* (LImages[0].img.Height / 3);

emptyX = LImages[8].rcDst.X;

emptyY = LImages[8].rcDst.Y;

}

void Shuffle()

{

int Z = 0;

int RandPos;

Random rr = new Random();

for (int i = 0; i < LImages.Count - 1; i++)

{

//rcSrc X

RandPos = rr.Next(0, 8);

CActorImage ptrav = LImages[i];

Z = ptrav.rcSrc.X;

ptrav.rcSrc.X = LImages[RandPos].rcSrc.X;

LImages[RandPos].rcSrc.X = Z;

//rcSrc Y

Z = 0;

Z = ptrav.rcSrc.Y;

ptrav.rcSrc.Y = LImages[RandPos].rcSrc.Y;

LImages[RandPos].rcSrc.Y = Z;

}

}

bool isClick(CActorImage ptrav, int xMouse, int yMouse)

{

if (xMouse >= ptrav.rcDst.X && xMouse <= ptrav.rcDst.X + ptrav.rcDst.Width && yMouse >= ptrav.rcDst.Y && yMouse <= ptrav.rcDst.Y + ptrav.rcDst.Height)

{

return true;

}

return false;

}

private bool IsAdjacent(CActorImage image, int emptyX, int emptyY)

{

int w = image.img.Width / 3;

int h = image.img.Height / 3;

if ((emptyX == image.rcDst.X + w || emptyX == image.rcDst.X - w) && emptyY == image.rcDst.Y)

{

return true;

}

if ((emptyY == image.rcDst.Y + h || emptyY == image.rcDst.Y - h) && emptyX == image.rcDst.X)

{

return true;

}

return false;

}

bool CheckWin()

{

for (int i = 0; i < LImages.Count; i++)

{

int correctX = (i % 3) \* (LImages[i].img.Width / 3);

int correctY = (i / 3) \* (LImages[i].img.Height / 3);

if (LImages[i].rcDst.X != correctX || LImages[i].rcDst.Y != correctY)

{

return false;

}

}

return true;

}

void DrawScene(Graphics g)

{

g.Clear(Color.Black);

g.FillRectangle(new SolidBrush(Color.Maroon), 0, 0, fullW - (fullW % 3), fullH);

for (int i = 0; i < LImages.Count - 1; i++)

{

g.DrawImage(LImages[i].img, LImages[i].rcDst, LImages[i].rcSrc, GraphicsUnit.Pixel);

}

}

void DrawDubb(Graphics g)

{

Graphics g2 = Graphics.FromImage(off);

DrawScene(g2);

g.DrawImage(off, 0, 0);

}

}

}