Problem 19

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\_19

{

public partial class Form1 : Form

{

Bitmap off;

Timer tt = new Timer();

public class CActorHelicopter

{

public int X, Y;

public Bitmap im;

}

public class CActorLand

{

public int X, Y;

public Bitmap im;

public int dirX;

}

public class CActorBird

{

public int X, Y;

public List<Bitmap> imgs;

public int iFrame;

public int dirX;

public int pos;

}

List<CActorHelicopter> LHelicopter = new List<CActorHelicopter>();

List<CActorBird> LBirdsHeli = new List<CActorBird>();

List<CActorBird> LBirdsLand = new List<CActorBird>();

List<CActorLand> LLandsMoving = new List<CActorLand>();

List<CActorLand> LLandsStable = new List<CActorLand>();

int near = -1;

bool upKey = false;

bool leftKey = false;

bool rightKey = false;

public Form1()

{

this.WindowState = FormWindowState.Maximized;

this.Load += Form1\_Load;

this.Paint += Form1\_Paint;

this.KeyDown += Form1\_KeyDown;

this.KeyUp += Form1\_KeyUp;

tt.Interval = 100;

tt.Start();

tt.Tick += Tt\_Tick;

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Up:

upKey = true;

break;

case Keys.Left:

leftKey = true;

break;

case Keys.Right:

rightKey = true;

break;

case Keys.Space:

if (near != -1 && near < LBirdsLand.Count)

{

CActorBird pnn = new CActorBird();

pnn.X = LBirdsLand[near].X;

pnn.Y = LHelicopter[0].Y + 80;

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

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

{

Bitmap im = new Bitmap("bird2.bmp");

im.MakeTransparent();

pnn.imgs.Add(im);

}

pnn.iFrame = 1;

LBirdsHeli.Add(pnn);

LBirdsLand.RemoveAt(near);

}

break;

}

}

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Up:

upKey = false;

break;

case Keys.Left:

leftKey = false;

break;

case Keys.Right:

rightKey = false;

break;

}

}

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

{

MoveLandsandBirds();

if (upKey == false)

{

MoveHelicopter();

}

else

{

LHelicopter[0].Y -= 8;

if (LBirdsHeli.Count > 0)

{

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

{

LBirdsHeli[i].Y -= 8;

}

}

}

if (leftKey)

{

LHelicopter[0].X -= 3;

if (LBirdsHeli.Count > 0)

{

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

{

LBirdsHeli[i].X -= 3;

}

}

}

if (rightKey)

{

LHelicopter[0].X += 3;

if (LBirdsHeli.Count > 0)

{

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

{

LBirdsHeli[i].X += 3;

}

}

}

ChangeBird();

DrawDubb(this.CreateGraphics());

}

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

{

DrawDubb(e.Graphics);

}

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

{

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

CreateHelicopter();

CreateLandsMoving();

CreateLandsStable();

CreateBirdsLand();

}

void CreateHelicopter()

{

CActorHelicopter pnn = new CActorHelicopter();

pnn.im = new Bitmap("helicopter.png");

pnn.X = this.ClientSize.Width - this.ClientSize.Width / 2;

pnn.Y = 200;

LHelicopter.Add(pnn);

}

void MoveHelicopter()

{

LHelicopter[0].Y += 3;

if (LBirdsHeli.Count > 0)

{

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

{

LBirdsHeli[i].Y += 3;

}

}

}

void CreateLandsMoving()

{

Random rr = new Random();

int xLands = 400;

for (int i = 0; i < 4; i++)

{

CActorLand pnn = new CActorLand();

pnn.im = new Bitmap("Land.bmp");

pnn.X = xLands;

pnn.Y = this.ClientSize.Height - 200;

if (i == 0 || i == 1)

{

pnn.dirX = -1;

}

else

{

pnn.dirX = 1;

}

xLands += 200;

LLandsMoving.Add(pnn);

}

}

void CreateLandsStable()

{

CActorLand pnn = new CActorLand();

pnn.im = new Bitmap("Land.bmp");

pnn.X = 0;

pnn.Y = this.ClientSize.Height - 200;

LLandsStable.Add(pnn);

pnn = new CActorLand();

pnn.im = new Bitmap("Land.bmp");

pnn.X = this.ClientSize.Width - 200; ;

pnn.Y = this.ClientSize.Height - 200;

LLandsStable.Add(pnn);

}

void CreateBirdsLand()

{

int xBirds = 410;

for (int j = 0; j < 4; j++)

{

CActorBird pnn = new CActorBird();

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

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

{

Bitmap im = new Bitmap("bird" + (i + 1) + ".bmp");

im.MakeTransparent();

pnn.imgs.Add(im);

}

pnn.X = xBirds;

pnn.Y = this.ClientSize.Height - 250;

if (j == 0 || j == 1)

{

pnn.dirX = -1;

}

else

{

pnn.dirX = 1;

}

pnn.pos = j;

xBirds += 200;

pnn.iFrame = 0;

LBirdsLand.Add(pnn);

}

}

void MoveLandsandBirds()

{

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

{

if (LLandsMoving[i].dirX == 1)

{

LLandsMoving[i].X += 5;

}

else

{

LLandsMoving[i].X -= 5;

}

}

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

{

if (LBirdsLand[i].dirX == 1)

{

LBirdsLand[i].X += 5;

}

else

{

LBirdsLand[i].X -= 5;

}

}

CollideLandsandBirds();

}

void CollideLandsandBirds()

{

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

{

if (i == 0)

{

if (LLandsMoving[i].X == 70)

{

LLandsMoving[i].dirX = 1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

else if (LLandsMoving[i].X + 70 == LLandsMoving[i + 1].X)

{

LLandsMoving[i].dirX = -1;

LLandsMoving[i + 1].dirX = 1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

}

else if (i == LLandsMoving.Count - 1)

{

if (LLandsMoving[i].X + 70 >= this.ClientSize.Width - 200)

{

LLandsMoving[i].dirX = -1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

else if (LLandsMoving[i].X == LLandsMoving[i - 1].X + 70)

{

LLandsMoving[i].dirX = 1;

LLandsMoving[i - 1].dirX = -1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

}

else

{

if (LLandsMoving[i].X + 70 == LLandsMoving[i + 1].X)

{

LLandsMoving[i].dirX = -1;

LLandsMoving[i + 1].dirX = 1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

else if (LLandsMoving[i].X == LLandsMoving[i - 1].X + 70)

{

LLandsMoving[i].dirX = 1;

LLandsMoving[i - 1].dirX = -1;

for (int k = 0; k < LBirdsLand.Count; k++)

{

if (LBirdsLand[k].pos == i)

{

LBirdsLand[k].dirX = LLandsMoving[LBirdsLand[k].pos].dirX;

}

}

}

}

}

}

void ChangeBird()

{

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

{

int birdX = LBirdsLand[i].X + 25;

if (birdX >= LHelicopter[0].X && birdX <= LHelicopter[0].X + 80 &&

LBirdsLand[i].Y >= LHelicopter[0].Y - 100 && LBirdsLand[i].Y <= LHelicopter[0].Y + 100)

{

LBirdsLand[i].iFrame = 1;

near = i;

}

else

{

LBirdsLand[i].iFrame = 0;

}

}

}

void DrawScene(Graphics g)

{

g.Clear(Color.GreenYellow);

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

{

g.DrawImage(LHelicopter[i].im, LHelicopter[i].X, LHelicopter[i].Y, 80, 80);

}

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

{

int index = LBirdsLand[i].iFrame % LBirdsLand[i].imgs.Count;

if (index == 0)

g.DrawImage(LBirdsLand[i].imgs[index], LBirdsLand[i].X, LBirdsLand[i].Y, 50, 50);

else

g.DrawImage(LBirdsLand[i].imgs[index], LBirdsLand[i].X, LBirdsLand[i].Y, 40, 40);

}

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

{

int index = LBirdsHeli[i].iFrame % LBirdsHeli.Count;

g.DrawImage(LBirdsHeli[i].imgs[index], LBirdsHeli[i].X, LBirdsHeli[i].Y, 40, 40);

}

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

{

g.DrawImage(LLandsMoving[i].im, LLandsMoving[i].X, LLandsMoving[i].Y, 70, 50);

}

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

{

g.DrawImage(LLandsStable[i].im, LLandsStable[i].X, LLandsStable[i].Y, 70, 50);

}

}

void DrawDubb(Graphics g)

{

Graphics g2 = Graphics.FromImage(off);

DrawScene(g2);

g.DrawImage(off, 0, 0);

}

}

}

Problem 20

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Problem20

{

public partial class Form1 : Form

{

private class CActorJerry

{

public int X, Y;

public Bitmap img;

}

private class CActorTom

{

public int X, Y;

public Bitmap img;

}

private class Shapes

{

public int X1, Y1, X2, Y2;

public Color cl;

}

private class Character

{

public List<Shapes> shapes = new List<Shapes>();

public int X, Y, W, H;

public Color cl;

}

private Timer tt = new Timer();

private CActorTom tom = new CActorTom();

private CActorJerry jerry = new CActorJerry();

private List<Character> LColumns = new List<Character>();

private List<Character> LRows = new List<Character>();

private Bitmap off;

private bool isDrag = false;

private int xOld;

private int yOld;

private int startMove;

private Character pCharacter = new Character();

private Shapes pR1 = new Shapes();

private Shapes pR2 = new Shapes();

private Shapes pR3 = new Shapes();

public Form1()

{

this.WindowState = FormWindowState.Maximized;

this.Load += Form1\_Load;

this.Paint += Form1\_Paint;

this.KeyDown += Form1\_KeyDown;

this.MouseDown += Form1\_MouseDown;

this.MouseMove += Form1\_MouseMove;

this.MouseUp += Form1\_MouseUp;

tt.Tick += Tt\_Tick;

tt.Interval = 100;

tt.Start();

}

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

{

if (startMove == 1)

{

MoveTom();

}

DrawDubb(this.CreateGraphics());

}

private void MoveTom()

{

tom.Y += 2;

int tomHalfW = tom.X + tom.img.Width \* 1 / 2;

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

{

// right row

if (tomHalfW >= LRows[i].shapes[0].X1 - 5 && tomHalfW <= LRows[i].shapes[1].X1 + 5 && tom.Y + tom.img.Height \* 1 / 2 >= LRows[i].shapes[0].Y1 && tom.Y + tom.img.Height \* 1 / 2 <= LRows[i].shapes[0].Y1 + LRows[i].shapes[0].Y2)

{

tom.X = LRows[i].shapes[2].X1 - tom.img.Width \* 1 / 2;

tom.Y = LRows[i].shapes[2].Y1;

}

//left row

if (tomHalfW <= LRows[i].shapes[2].X1 + LRows[i].shapes[2].X2 + 5 && tomHalfW >= LRows[i].shapes[1].X2 - 5 && tom.Y + tom.img.Height \* 1 / 2 >= LRows[i].shapes[2].Y1 && tom.Y + tom.img.Height \* 1 / 2 <= LRows[i].shapes[2].Y1 + LRows[i].shapes[2].Y2)

{

tom.X = LRows[i].shapes[0].X1 - tom.img.Width \* 1 / 2;

tom.Y = LRows[i].shapes[0].Y1;

}

}

//reach end or jerry

if (tomHalfW >= jerry.X && tomHalfW <= jerry.X + jerry.img.Width && tom.Y >= jerry.Y)

{

tt.Stop();

MessageBox.Show("CONGRATS!!");

}

else if (tom.Y >= jerry.Y)

{

tt.Stop();

MessageBox.Show("oops, you lost :/");

}

}

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

{

isDrag = false;

pCharacter.shapes.Add(pR1);

pCharacter.shapes.Add(pR2);

pR3.X1 = e.X - 2;

pR3.Y1 = e.Y;

pR3.X2 = 10;

pR3.Y2 = 10;

pR3.cl = Color.Red;

pCharacter.shapes.Add(pR3);

LRows.Add(pCharacter);

pCharacter = null;

pR1 = null;

pR2 = null;

pR3 = null;

}

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

{

if (isDrag == true)

{

int dx = e.X - xOld;

int dy = e.Y - yOld;

//line

pR1.X1 = xOld;

pR1.Y1 = yOld;

pR1.X2 = 10;

pR1.Y2 = 10;

pR1.cl = Color.Red;

//ellipse

pR2.X1 = xOld + 5;

pR2.Y1 = yOld;

pR2.X2 = e.X;

pR2.Y2 = e.Y;

pR2.cl = Color.Green;

}

}

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

{

if (e.Button == MouseButtons.Left)

{

xOld = e.X;

yOld = e.Y;

pCharacter = new Character();

pR1 = new Shapes();

pR2 = new Shapes();

pR3 = new Shapes();

isDrag = true;

}

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.Space:

LColumns.Clear();

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

{

LRows[i].shapes.Clear();

}

LRows.Clear();

CreateColumn();

CreateActor();

CreateRows();

break;

case Keys.Enter:

startMove = 1;

break;

}

}

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

{

DrawDubb(e.Graphics);

}

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

{

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

CreateColumn();

CreateActor();

CreateRows();

}

private void CreateRows()

{

Random rr = new Random();

Random r = new Random();

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

{

int v = rr.Next(1, 3); //no.of rows in one column

for (int j = 0; j < v; j++)

{

int yy = r.Next(LColumns[i].Y + 2, LColumns[i].Y + LColumns[i].H - 2); //postion of y of each column

Character pnnR = new Character();

Shapes pShape1 = new Shapes();

pShape1.X1 = LColumns[i].X;

pShape1.Y1 = yy;

pShape1.X2 = 10;

pShape1.Y2 = 10;

pShape1.cl = Color.Blue;

pnnR.shapes.Add(pShape1);

Shapes pShape2 = new Shapes();

pShape2.X1 = LColumns[i].X + 5;

pShape2.Y1 = yy;

pShape2.X2 = LColumns[i].X + 100;

pShape2.Y2 = yy;

pShape2.cl = Color.Red;

pnnR.shapes.Add(pShape2);

Shapes pShape3 = new Shapes();

pShape3.X1 = LColumns[i].X + 100 + 2;

pShape3.Y1 = yy;

pShape3.X2 = 10;

pShape3.Y2 = 10;

pShape3.cl = Color.Blue;

pnnR.shapes.Add(pShape3);

LRows.Add(pnnR);

}

}

}

private void CreateActor()

{

tom.img = new Bitmap("tom.bmp");

tom.img.MakeTransparent(tom.img.GetPixel(0, 0));

Random r = new Random();

int v = r.Next(0, LColumns.Count());

tom.X = LColumns[v].X - tom.img.Width \* 1 / 2;

tom.Y = LColumns[v].Y - tom.img.Height + 5;

tom.dirX = 0;

jerry.img = new Bitmap("jerry.bmp");

jerry.img.MakeTransparent(jerry.img.GetPixel(0, 0));

Random rr = new Random();

int vv = r.Next(0, LColumns.Count());

jerry.X = LColumns[vv].X - jerry.img.Width \* 1 / 2;

jerry.Y = LColumns[vv].Y + LColumns[vv].H;

}

private void CreateColumn()

{

Random rr = new Random();

int v = rr.Next(2, 6);

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

{

Character pnn = new Character();

pnn.X = (this.ClientSize.Width \* 1 / 9) + (i \* 100);

pnn.Y = 80;

pnn.W = 5;

pnn.H = 420;

pnn.cl = Color.Black;

LColumns.Add(pnn);

}

}

private void DrawScene(Graphics g)

{

g.Clear(Color.YellowGreen);

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

{

g.DrawRectangle(new Pen(LColumns[0].cl), LColumns[i].X, LColumns[i].Y, LColumns[i].W, LColumns[i].H);

g.FillRectangle(new SolidBrush(LColumns[0].cl), LColumns[i].X, LColumns[i].Y, LColumns[i].W, LColumns[i].H);

}

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

{

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

{

if (j == 1) //row

{

g.DrawLine(new Pen(LRows[0].shapes[j].cl, 5), LRows[i].shapes[j].X1, LRows[i].shapes[j].Y1, LRows[i].shapes[j].X2, LRows[i].shapes[j].Y2);

}

else

{

//ellipses

g.DrawEllipse(new Pen(LRows[0].shapes[j].cl), LRows[i].shapes[j].X1, LRows[i].shapes[j].Y1, LRows[i].shapes[j].X2, LRows[i].shapes[j].Y2);

g.FillEllipse(new SolidBrush(LRows[0].shapes[j].cl), LRows[i].shapes[j].X1, LRows[i].shapes[j].Y1, LRows[i].shapes[j].X2, LRows[i].shapes[j].Y2);

}

if (isDrag == true)

{

if (j == 1)

{

g.DrawLine(new Pen(pR2.cl, 5), pR2.X1, pR2.Y1, pR2.X2, pR2.Y2);

}

else

{

if (j == 0) //first ellipse

{

g.DrawEllipse(new Pen(pR1.cl), pR1.X1, pR1.Y1, pR1.X2, pR1.Y2);

g.FillEllipse(new SolidBrush(pR1.cl), pR1.X1, pR1.Y1, pR1.X2, pR1.Y2);

}

}

}

}

}

g.DrawImage(tom.img, tom.X, tom.Y);

g.DrawImage(jerry.img, jerry.X, jerry.Y);

}

private void DrawDubb(Graphics g)

{

Graphics g2 = Graphics.FromImage(off);

DrawScene(g2);

g.DrawImage(off, 0, 0);

}

}

}