Университет ИТМО

кафедра ИПМ

Лабораторная работа №1

Вариант 3

«Анимация битового образа»

Работу выполнил

Студент 4 курса

Группы № P3418

Журавлев Виталий

Санкт-Петербург

2015 год

**Задание**

 Написать приложение Windows, которое выполняет анимацию

битового образа.

 **Этапы выполнения работы.**

1) Создать главное окно. Заголовок содержит ФИО, гр., вар.

2) Создать меню вида: | Begin | Animate | Stop | About | Quit

 │Choose │

 │Show picture│

 Пункты меню "Show picture", "Animate" и "Stop" запрещены.

 По Quit приложение завершается.

 По About открывается окно с информацией о теме разработки.

3) Пo Choose открывается диалоговое окно вида:



 где 1,2,3,4 - окна-pедактоpы для ввода X и Y составляющих кооpдинат границ

 движения объекта ( 1 и 2 начальная точка, 3 и 4 - конечная) в пикселах,

 5 - окно-список для ввода скоpости движения объекта(в условных единицах),

 6 - статическое окно для вывода текущего размера рабочей области главного

 окна в пикселах,

 OK, Cancel - кнопки типа BS\_PUSHBUTTON.

 Если координаты границ движения объекта не принадлежат рабочей области окна,

 должно быть выдано соответствующее сообщение с дальнейшей установкой фокуса

 ввода на 1-е окно с неверным значением.

 При инициализации диалога фокус ввода должно иметь окно 1.

 При первом выборе этого пункта меню окна 1,2,3 и 4 должны содержать

 значения, соответствующие всей рабочей области главного окна.

 При выборе "OK" (и верных значениях в 1,2,3 и 4) выбор запоминается,

 диалог завершается и разрешается пункт меню "Show picture".

 При этом в главном окне отрисовывается контур прямоугольника с координатами,

 заданными в 1,2,3,4.

 Если значения в 1,2,3 и 4 не принадлежат рабочей области главного

 окна, выдать сообщение об ошибке и установить фокус ввода на 1-е окно с

 неверными значениями без завершения диалога.

 При выборе "Cancel" диалог завершается,выбор игнорируется и пункт меню

 "Show picture" остается в предыдущем состоянии.

4) По "Show picture" в центре указанной в 1,2,3,4 области главного окна

 рисуется объект,состоящий из прямоугольника размером 5мм на 10мм синего

 цвета и в него вписывается круг диаметром 5мм желтого цвета.

 Данный объект должен отрисовываться програмно, а не браться из файла.

 Он запоминается в виде битового образа.

 Разрешается пункт меню "Animate".

 При повторных вхождениях в этот пункт меню предыдущий объект д.б. уничтожен.

5) По "Animate" разрешается пункт меню "Stop" и объект перемещается в начальную

 точку и начинает перемещаться вправо до достижения правой границы,далее вниз

 до достижения нижней границы, затем влево до левой границы,затем вверх до

 верхней границы и так далее с заданной скоpостью.

 Запрещаются пункты меню "Animate" и "Begin".

6) По Stop - объект пpекpащает движение.

 Разрешаются пункты меню "Animate" и "Begin".

 Окно должно располагаться в центре экрана и иметь размеры по 0.75 от ширины

 и высоты экрана соответственно.

 Окно не должно позволять изменять свои размеры.

**Основные элемента кода программы**

**public partial class MainForm : Form**

 {

 public static double mmInInch = 0.0393701;

 public static double ppiDensity = 96;

 private ChooseForm chooseDialog;

 private Worker workerObject;

 private Bitmap bitmap = null;

 private bool isShowBitmap = false;

 protected float curX;

 protected float curY;

 public MainForm()

 {

 InitializeComponent();

 this.Size = new System.Drawing.Size((int)(System.Windows.Forms.Screen.PrimaryScreen.WorkingArea.Width \* 0.75),

 (int)(System.Windows.Forms.Screen.PrimaryScreen.WorkingArea.Height \* 0.75));

 this.DoubleBuffered = true;

 }

 private void showPictureToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 if (bitmap != null)

 bitmap.Dispose();

 createBitmap();

 curX = this.ClientSize.Width / 2 - bitmap.Width / 2;

 curY = this.ClientSize.Height / 2 - bitmap.Height / 2;

 animateToolStripMenuItem.Enabled = true;

 this.Refresh();

 }

 private void aboutToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 AboutBox aboutBox = new AboutBox();

 aboutBox.Show();

 }

 private void quitToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 Application.Exit();

 }

 private void chooseToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 DialogResult dr = new DialogResult();

 chooseDialog = new ChooseForm(this.ClientSize.Height - menuStrip1.Height, this.ClientSize.Width);

 dr = chooseDialog.ShowDialog();

 if (dr == DialogResult.OK)

 {

 this.showPictureToolStripMenuItem.Enabled = true;

 isShowBitmap = true;

 }

 }

 private void MainForm\_SizeChanged(object sender, EventArgs e)

 {

 if (chooseDialog != null && !chooseDialog.IsDisposed)

 chooseDialog.SetWindowSizeText(this.ClientSize.Height, this.ClientSize.Width);

 }

 private void MainForm\_Paint(object sender, System.Windows.Forms.PaintEventArgs e)

 {

 if (bitmap != null && isShowBitmap)

 {

 e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

 e.Graphics.DrawImage(bitmap, curX, curY);

 }

 }

 private void createBitmap()

 {

 this.bitmap = new Bitmap((int)(10 \* mmInInch \* ppiDensity), (int)(5 \* mmInInch \* ppiDensity));

 Graphics graphics = Graphics.FromImage(bitmap);

 graphics.FillRectangle(Brushes.Blue, 0, 0, bitmap.Width, bitmap.Height);

 graphics.FillEllipse(Brushes.Yellow, (float)(0.25 \* bitmap.Width), 0, bitmap.Height, bitmap.Height);

 graphics.Dispose();

 }

 private void animateToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 stopToolStripMenuItem.Enabled = true;

 beginMenuItem.Enabled = false;

 animateToolStripMenuItem.Enabled = false;

 curX = Properties.Settings.Default.startWidth;

 curY = Properties.Settings.Default.startHeight + menuStrip1.Height;

 // Create the thread object. This does not start the thread.

 workerObject = new Worker(this);

 Thread workerThread = new Thread(workerObject.Animate);

 workerThread.Start();

 this.Refresh();

 }

 private void stopToolStripMenuItem\_Click(object sender, EventArgs e)

 {

 workerObject.RequestStop();

 beginMenuItem.Enabled = true;

 animateToolStripMenuItem.Enabled = true;

 stopToolStripMenuItem.Enabled = false;

 }

 public void Redraw()

 {

 this.Invoke((MethodInvoker)delegate

 {

 if (!this.IsDisposed)

 {

 this.Refresh();

 }

 });

 }

 public class Worker

 {

 private volatile bool stopped;

 private MainForm mf;

 public Worker(MainForm mf)

 {

 this.mf = mf;

 }

 public void Animate()

 {

 int maxWidth = Math.Max(Properties.Settings.Default.endWidth, Properties.Settings.Default.startWidth);

 int minWidth = Math.Min(Properties.Settings.Default.endWidth, Properties.Settings.Default.startWidth);

 if (maxWidth - minWidth > mf.bitmap.Width)

 maxWidth -= mf.bitmap.Width;

 int maxHeight = Math.Max(Properties.Settings.Default.endHeight, Properties.Settings.Default.startHeight) + mf.menuStrip1.Height;

 int minHeight = Math.Min(Properties.Settings.Default.endHeight, Properties.Settings.Default.startHeight) + mf.menuStrip1.Height;

 if (maxHeight - minHeight > mf.bitmap.Height)

 maxHeight -= mf.bitmap.Height;

 while (!stopped)

 {

 while (!stopped && maxWidth - mf.curX > 0)

 {

 mf.curX += 1;

 mf.Redraw();

 Thread.Sleep(40 / Properties.Settings.Default.speed + 1 < 1 ? 1 : Properties.Settings.Default.speed + 1);

 }

 while (!stopped && maxHeight - mf.curY > 0)

 {

 mf.curY += 1;

 mf.Redraw();

 Thread.Sleep(40 / Properties.Settings.Default.speed + 1);

 }

 while (!stopped && minWidth < mf.curX)

 {

 mf.curX -= 1;

 mf.Redraw();

 Thread.Sleep(40 / Properties.Settings.Default.speed + 1);

 }

 while (!stopped && minHeight < mf.curY)

 {

 mf.curY -= 1;

 mf.Redraw();

 Thread.Sleep(40 / Properties.Settings.Default.speed + 1);

 }

 }

 }

 public void RequestStop()

 {

 stopped = true;

 }

 }

 }

**partial class ChooseForm : Form**

 {

 public ChooseForm(int mainHeight, int mainWidth)

 {

 this.mainHeight = mainHeight;

 this.mainWidth = mainWidth;

 InitializeComponent();

 SetValues();

 }

 private void SetValues()

 {

 if (Properties.Settings.Default.startHeight != -1)

 {

 this.textBox1.Text = Properties.Settings.Default.startHeight.ToString();

 }

 else

 this.textBox1.Text = "0";

 if (Properties.Settings.Default.startWidth != -1)

 {

 this.textBox2.Text = Properties.Settings.Default.startWidth.ToString();

 }

 else

 this.textBox2.Text = "0";

 if (Properties.Settings.Default.endHeight != -1)

 {

 this.textBox3.Text = Properties.Settings.Default.endHeight.ToString();

 }

 else if (this.mainHeight > 0)

 this.textBox3.Text = this.mainHeight.ToString();

 else

 this.textBox3.Text = "0";

 if (Properties.Settings.Default.endWidth != -1)

 {

 this.textBox4.Text = Properties.Settings.Default.endWidth.ToString();

 }

 else if (this.mainWidth > 0)

 this.textBox4.Text = this.mainWidth.ToString();

 else

 this.textBox4.Text = "0";

 SetWindowSizeText(mainHeight, mainWidth);

 if (Properties.Settings.Default.speed >= 0)

 this.listBox1.SetSelected(Properties.Settings.Default.speed, true);

 else

 this.listBox1.SetSelected(0, true);

 }

 public void SetWindowSizeText(int mainHeight, int mainWidth)

 {

 this.mainHeight = mainHeight;

 this.mainWidth = mainWidth;

 this.textBox5.Text = String.Format("{0} x {1}", mainHeight, mainWidth);

 }

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

 {

 if (CheckValues())

 {

 Properties.Settings.Default.startHeight = int.Parse(textBox1.Text);

 Properties.Settings.Default.startWidth = int.Parse(textBox2.Text);

 Properties.Settings.Default.endHeight = int.Parse(textBox3.Text);

 Properties.Settings.Default.endWidth = int.Parse(textBox4.Text);

 Properties.Settings.Default.speed = listBox1.SelectedIndex;

 this.Close();

 }

 }

 private bool CheckValues()

 {

 bool result = true;

 if (textBox4.Text.Length == 0 || int.Parse(textBox4.Text) < 0 || int.Parse(textBox4.Text) > mainWidth)

 {

 textBox4.Clear();

 valueErrorProvider.SetError(textBox4, "Wrong value!");

 textBox4.Focus();

 result = false;

 }

 if (textBox3.Text.Length == 0 || int.Parse(textBox3.Text) < 0 || int.Parse(textBox3.Text) > mainHeight)

 {

 textBox3.Clear();

 valueErrorProvider.SetError(textBox3, "Wrong value!");

 textBox3.Focus();

 result = false;

 }

 if (textBox2.Text.Length == 0 || int.Parse(textBox2.Text) < 0 || int.Parse(textBox2.Text) > mainWidth)

 {

 textBox2.Clear();

 valueErrorProvider.SetError(textBox2, "Wrong value!");

 textBox2.Focus();

 result = false;

 }

 if (textBox1.Text.Length == 0 || int.Parse(textBox1.Text) < 0 || int.Parse(textBox1.Text) > mainHeight)

 {

 textBox1.Clear();

 valueErrorProvider.SetError(textBox1, "Wrong value!");

 textBox1.Focus();

 result = false;

 }

 return result;

 }

 private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

 {

 valueErrorProvider.Clear();

 e.Handled = isNumberChar(e.KeyChar);

 }

 private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

 {

 valueErrorProvider.Clear();

 e.Handled = isNumberChar(e.KeyChar);

 }

 private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

 {

 valueErrorProvider.Clear();

 e.Handled = isNumberChar(e.KeyChar);

 }

 private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

 {

 valueErrorProvider.Clear();

 e.Handled = isNumberChar(e.KeyChar);

 }

 private bool isNumberChar(Char ch)

 {

 return !char.IsControl(ch) && !char.IsDigit(ch);

 }

 private void button2\_Click(object sender, EventArgs e)

 {

 this.Close();

 }

 }

**Вывод**

 В ходе выполнения лабораторной работы были изучены способы рисования и анимирования в C#, способы работы с данными внутри программы, а также взаимодействие с диалоговыми окнами. Была создана программа, показывающая анимацию битового изображения.