C#程序设计语言与

.Net框架基础（双语）

**实**

**验**

**报**

**告**

**学 院 软件学院**

**年 级 2017级**

**班 级 一班**

**学 号 3017218060**

**姓 名 刘杭学**

**2019 年 4 月 4 日**



**C#程序设计语言与.Net框架基础（双语）**

**上机实验报告**

实验二 题目：Windows Form实现MIDI音乐文件的播放APP

**学院名称 软件学院**

**专 业 软件工程**

**学生姓名 刘杭学**

**学 号 3017218060**

**年 级 2017级**

**班 级 一班**

**时 间 2019年4月4日**

目 录

[实验名称 1](#_Toc509419319)

[实验目的 1](#_Toc509419320)

[实验内容 1](#_Toc509419321)

[实验要求 4](#_Toc509419322)

[实验分析 4](#_Toc509419323)

[实验结论及心得体会 5](#_Toc509419324)

## 实验名称

Windows Form实现MIDI音乐文件的播放APP

## 实验目的

1) 理解和掌握基于Windows Form的APP应用程序开发

2) 学习使用MIDI Toolkit完成Midi音乐文件的播放

3) 理解C# MIDI Toolkit内部的类、event、delegate构成机制

4) 理解基于Windows Form的event的GUI界面开发方法

## 实验内容

1) 使用C# MIDI Toolkit提供的源程序，在Visual Studio中建立相应的解决方案。

2) 能够成功编译C# MIDI Toolkit提供的演示程序。并能正常播放MIDI文件。

3) 理解演示程序的内部工作机制: 参照C# MIDI Toolkit文章内容，理解Event/Delegate方式实现的模块间的耦合机制，各种类的继承关系等。

4) 对GUI界面中的控件大小、位置进行完善，使之能够随APP界面大小自动调整其自身大小。需要使用相应的Event完成此项工作。

5) 其他GUI界面的用户体验提升：由同学们自己提出创新功能并实现。

## 实验要求

参实验内容完善详细的实验步骤，每位同学独立或两位同学结对编程完成实验。

## 实验分析

1. 实验结果

1.1项目名称：实现空间大小的自适应

1.2操作步骤：

1. 新建相关类AutoSizeFormClass
2. 生成对象AutoSizeFormClass asc = new AutoSizeFormClass();
3. 新增事件

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

{

asc.controlAutoSize(this);

}

1. AutoSizeFormClass相关代码

using System.Collections.Generic;

using System.Windows.Forms;

class AutoSizeFormClass

{

//(1).声明结构,只记录窗体和其控件的初始位置和大小。

public struct controlRect

{

public int Left;

public int Top;

public int Width;

public int Height;

}

//(2).声明 1个对象

//注意这里不能使用控件列表记录 List nCtrl;，因为控件的关联性，记录的始终是当前的大小。

// public List oldCtrl= new List();//这里将西文的大于小于号都过滤掉了，只能改为中文的，使用中要改回西文

public List<controlRect> oldCtrl = new List<controlRect>();

int ctrlNo = 0;//1;

//(3). 创建两个函数

//(3.1)记录窗体和其控件的初始位置和大小,

public void controllInitializeSize(Control mForm)

{

controlRect cR;

cR.Left = mForm.Left; cR.Top = mForm.Top; cR.Width = mForm.Width; cR.Height = mForm.Height;

oldCtrl.Add(cR);//第一个为"窗体本身",只加入一次即可

AddControl(mForm);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

//this.WindowState = (System.Windows.Forms.FormWindowState)(2);//记录完控件的初始位置和大小后，再最大化

//0 - Normalize , 1 - Minimize,2- Maximize

}

private void AddControl(Control ctl)

{

foreach (Control c in ctl.Controls)

{ //\*\*放在这里，是先记录控件的子控件，后记录控件本身

//if (c.Controls.Count > 0)

// AddControl(c);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

controlRect objCtrl;

objCtrl.Left = c.Left; objCtrl.Top = c.Top; objCtrl.Width = c.Width; objCtrl.Height = c.Height;

oldCtrl.Add(objCtrl);

//\*\*放在这里，是先记录控件本身，后记录控件的子控件

if (c.Controls.Count > 0)

AddControl(c);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

}

}

//(3.2)控件自适应大小,

public void controlAutoSize(Control mForm)

{

if (ctrlNo == 0)

{ //\*如果在窗体的Form1\_Load中，记录控件原始的大小和位置，正常没有问题，但要加入皮肤就会出现问题，因为有些控件如dataGridView的的子控件还没有完成，个数少

//\*要在窗体的Form1\_SizeChanged中，第一次改变大小时，记录控件原始的大小和位置,这里所有控件的子控件都已经形成

controlRect cR;

// cR.Left = mForm.Left; cR.Top = mForm.Top; cR.Width = mForm.Width; cR.Height = mForm.Height;

cR.Left = 0; cR.Top = 0; cR.Width = mForm.PreferredSize.Width; cR.Height = mForm.PreferredSize.Height;

oldCtrl.Add(cR);//第一个为"窗体本身",只加入一次即可

AddControl(mForm);//窗体内其余控件可能嵌套其它控件(比如panel),故单独抽出以便递归调用

}

float wScale = (float)mForm.Width / (float)oldCtrl[0].Width;//新旧窗体之间的比例，与最早的旧窗体

float hScale = (float)mForm.Height / (float)oldCtrl[0].Height;//.Height;

ctrlNo = 1;//进入=1，第0个为窗体本身,窗体内的控件,从序号1开始

AutoScaleControl(mForm, wScale, hScale);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

}

private void AutoScaleControl(Control ctl, float wScale, float hScale)

{

int ctrLeft0, ctrTop0, ctrWidth0, ctrHeight0;

//int ctrlNo = 1;//第1个是窗体自身的 Left,Top,Width,Height，所以窗体控件从ctrlNo=1开始

foreach (Control c in ctl.Controls)

{ //\*\*放在这里，是先缩放控件的子控件，后缩放控件本身

//if (c.Controls.Count > 0)

// AutoScaleControl(c, wScale, hScale);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

ctrLeft0 = oldCtrl[ctrlNo].Left;

ctrTop0 = oldCtrl[ctrlNo].Top;

ctrWidth0 = oldCtrl[ctrlNo].Width;

ctrHeight0 = oldCtrl[ctrlNo].Height;

//c.Left = (int)((ctrLeft0 - wLeft0) \* wScale) + wLeft1;//新旧控件之间的线性比例

//c.Top = (int)((ctrTop0 - wTop0) \* h) + wTop1;

c.Left = (int)((ctrLeft0) \* wScale);//新旧控件之间的线性比例。控件位置只相对于窗体，所以不能加 + wLeft1

c.Top = (int)((ctrTop0) \* hScale);//

c.Width = (int)(ctrWidth0 \* wScale);//只与最初的大小相关，所以不能与现在的宽度相乘 (int)(c.Width \* w);

c.Height = (int)(ctrHeight0 \* hScale);//

ctrlNo++;//累加序号

//\*\*放在这里，是先缩放控件本身，后缩放控件的子控件

if (c.Controls.Count > 0)

AutoScaleControl(c, wScale, hScale);//窗体内其余控件还可能嵌套控件(比如panel),要单独抽出,因为要递归调用

if (ctl is DataGridView)

{

DataGridView dgv = ctl as DataGridView;

Cursor.Current = Cursors.WaitCursor;

int widths = 0;

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

{

dgv.AutoResizeColumn(i, DataGridViewAutoSizeColumnMode.AllCells); // 自动调整列宽

widths += dgv.Columns[i].Width; // 计算调整列后单元列的宽度和

}

if (widths >= ctl.Size.Width) // 如果调整列的宽度大于设定列宽

dgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.DisplayedCells; // 调整列的模式 自动

else

dgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill; // 如果小于 则填充

Cursor.Current = Cursors.Default;

}

}

}

}

1.3实际结果描述、结论：

成功实现了控件大小随页面的自适应变化

2.1项目名称： 添加listBox实现播放内容的可视化以及拖拽播放

2.2操作步骤：

1. 拖拽控件增加listBox
2. 新增事件

private void Form1\_DragEnter(object sender, DragEventArgs e) // 拖拽播放

{

string path = ((System.Array)e.Data.GetData(DataFormats.FileDrop)).GetValue(0).ToString();

listBox1.Items.Add(path);

}

2.3实际结果描述、结论：

1. 成功实现了添加listBox实现播放内容的可视化以及拖拽播放
2. listBox 每一行的内容一起组成了一个集合

3.1项目名称：实现音乐的单曲循环，顺序播放，和随机播放

3.2操作步骤：

1. 拖拽控件添加
2. 新增代码

private void HandlePlayingCompleted(object sender, EventArgs e)

{

if (AutoLoop)

{

string path = listBox1.SelectedItem.ToString();

Open(path); // 选中播放

}

else if(InOrder)

{

listBox1.SelectedIndex = (listBox1.SelectedIndex + 1) % (listBox1.Items.Count); //跳到下一个

}else

{

int i = listBox1.SelectedIndex;

Random rd = new Random();

int r = rd.Next(0,listBox1.Items.Count);

if(r==i)

{

string path = listBox1.SelectedItem.ToString();

Open(path); // 选中播放

}

else

{

listBox1.SelectedIndex = r;

}

}

}

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

{

//listBox1.SelectionMode =;

rb1.Checked = true; // 默认单曲循环

Control.CheckForIllegalCrossThreadCalls = false;

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

string path = listBox1.SelectedItem.ToString();

Open(path); // 选中播放

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

AutoLoop = true;

InOrder = false;

randomOrder = false;

}

private void rb2\_CheckedChanged(object sender, EventArgs e)

{

AutoLoop = false;

InOrder = true;

randomOrder = false;

}

private void rb3\_CheckedChanged(object sender, EventArgs e)

{

AutoLoop = false;

InOrder = false;

randomOrder = true;

}

}

3.3实际结果描述、结论：

实现了音乐的单曲循环，顺序播放，随机播放

了解了线程保护，和跨线程调用的相关知识

4.1项目名称： 增加退出按钮

4.2操作步骤：

private void quitBtn\_Click(object sender, EventArgs e) //退出窗体

{

DialogResult result= MessageBox.Show(this, "Are you sure you want to quit the program? ", "Closing prompt ", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (result == DialogResult.OK)

{

//this.Close();

System.Environment.Exit(0);//这是最彻底的退出方式，不管什么线程都被强制退出，把程序结束的很干净。

}

}

4.3实际结果描述、结论：

实现了退出按钮

## 实验结论及心得体会

1) 理解和掌握了基于Windows Form的APP应用程序开发

2) 学会使用MIDI Toolkit完成Midi音乐文件的播放

3) 理解了C# MIDI Toolkit内部的类、event、delegate构成机制

4) 理解了基于Windows Form的event的GUI界面开发方法

5) 掌握了winform中各种控件的使用方法

6) 了解并掌握了C#中的线程保护和跨线程调用的相关知识。