

C 博客

登录 | 注册

Q

≡

永远即等待的专栏 记录自己的点点滴滴

目录视图

摘要视图

RSS 订阅

个人资料



leehong2005

+ 加关注

发私信

访问：764738次

积分：6283

等级：

BLOG-6

排名：第2858名

原创：102篇 转载：1篇

译文：0篇 评论：639条

文章搜索

Q

文章分类

Android (45)

C/C++ (11)

Windows (6)

VC++ (23)

COM (15)

团队管理 (1)

设计 (5)

文章存档

2014年03月 (1)

2014年01月 (2)

2013年10月 (1)

2013年09月 (1)

2013年07月 (1)

↓ 展开

阅读排行

Android 下拉刷新框架实

(105211)

Android 关于OOM的解决

(51711)

Android debug.keystore

(48725)

Android WebView的Js对

(48295)

Android 实现 WheelView

(47892)

Android java.lang.NoCl

(42516)

💡 移动信息安全的漏洞和逆向原理

📰 程序员11月书讯，评论得书啦

🔗 Get IT技能知识库，50个领域一键直达

原 C#基本图像处理

2013-02-26 12:33 7606人阅读 评论(1) 收藏 举报

分类： VC++ (22)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

最近没事，有网上看到一篇关于图像处理的文章，觉得很好，结合它上面的原理，自己写了一个C#图像处理的例子。这个DEMO的界面的有两个PictureBox控件，用来显示图片，一个是源图片，一个是经过转换的目标图片，UI下面部分有一些按钮，每个按钮实现一个转换功能。这个DEMO允许用户拖一张图片到源PictureBox中，然后通过这些功能按钮实现图片的效果转换。这些功能有把图片变成黑白、底片、浮雕、锐化、柔化等效果。首先来一张运行效果图：



这个DEMO主要有以下几个机能点：

☐ 图像处理

☐ PictureBox的拖拽。

☐ 计算处理时间

☐ 对图像进行缩放处理

1.图像处理

首先说明一点，图像处理的算法不是我自己的想出来的，也没有必要去想，网上调查一下，很多的。所以算法是网是找的。在些声明一下。

☐ 1.1 黑白效果

原理: 彩色图像处理成黑白效果通常有3种算法：

(1).最大值法: 使每个像素点的 R, G, B 值等于原像素点的 RGB (颜色值) 中最大的一个；

(2).平均值法: 使用每个像素点的 R,G,B值等于原像素点的RGB值的平均值；

(3).加权平均值法: 对每个像素点的 R, G, B值进行加权，R, G, B的系数分别是0.7, 0.2, 0.1。

自认为第三种的效果是最好的。

快速回复

http://blog.csdn.net/leehong2005/article/details/8613145

1/10

- Android ListView分组和悬
- (37234)
- Android 程序框架设计
- (36724)
- Android gallery 3D效果
- (30544)
- Android 数据库升级解决
- (22677)

评论排行

- Android 下拉刷新框架实
- (225)
- Android 画图板程序实例
- (76)
- Android 实现 WheelView
- (75)
- Android 程序框架设计
- (41)
- Android WebView的Js对
- (33)
- Android gallery 3D效果
- (32)
- Android 吸入动画效果详
- (25)
- Android ListView分组和悬
- (18)
- Android 数据库升级解决
- (15)
- Android 异步链式调用设
- (12)

推荐文章

- \* 程序员10月书讯, 评论得书
- \* Android中Xposed框架篇---修改系统位置信息实现自身隐藏功能
- \* Chromium插件（Plugin）模块（Module）加载过程分析
- \* Android TV开发总结--构建一个TV app的直播节目实例
- \* 架构设计：系统存储--MySQL简单主从方案及暴露的问题

最新评论

- Android 下拉刷新框架实现  
dev\_test: 很好, 可以直接使用, 非常感谢!
- Android gallery 3D效果  
yibin12388: mark下
- Android WebView的Js对象注入:  
xiaomao5200: 需要过滤掉Object类的方法这一段, 请问如何过滤掉这些方法? 还有既然4.2以下版本采用的方案, 已经...
- Android java.lang.NoClassDef  
flashyhx: 太感动了, 已解决
- Android WebView的Js对象注入:  
xiaonaihe: 拿到项目上用了, 非常感谢博主提供的方案!
- Android 画图板程序实例 (Sketc  
qq\_36023234: 不得不告诉你一个残酷的事实 你的APP已经不纯在了
- Android gallery 3D效果  
android阿杜: 4.0之后, google就不支持这个控件了, 展示的demo中, 中间的图片会是歪的。而且还不支持set...
- Android gallery 3D效果  
android阿杜: 4.0之后, google就不支持这个控件了, 展示的demo中, 中间的图片会是歪的。而且还不支持set...
- C#基本图像处理  
tianyawp123: 那请问这些代码是怎么组起来的? 是在不同的文件夹里吗
- Android 程序框架设计  
wlh8484: return convertView; ---  
-----》return ...

1.2 底片效果

原理: GetPixel方法获得每一点像素的值, 然后再使用SetPixel方法将取反后的颜色值设置到对应的点。

1.3 锐化效果

原理:突出显示颜色值大(即形成形体边缘)的像素点。

1.4 浮雕效果

原理: 对图像像素点的像素值分别与相邻像素点的像素值相减后加上128, 然后将其作为新的像素点的值。

1.5 柔化效果

原理: 当前像素点与周围像素点的颜色差距较大时取其平均值。

返回顶部

2.PictureBox的拖拽

拖拽是写在UserControlPictureBox类中, 该类继承于UserControl, 里面有一个PicttrueBox, 相当于把图片显示, 缩放, 拖拽封装了。

C#的拖拽还是很简单的。主要用到DragEnter和DragDrop事件和DoDragDrop方法。

DragEnter

在拖拽源被拖入到拖拽目标时触发, 在这个事件处理函数中, 要做的事情就是设置DragEventArgs 对象的Effect, 这是一个DragDropEffects枚举值。具体请参见MSDN。

DragDrop

在释放鼠标并且鼠标拖拽目标之内在时发生。这里面可以接受拖拽的数据。

DoDragDrop

这个函数表示开始一个拖拽事件, 一般是在MouseDown或者MouseMove中调用这个函数, 这个函数会阻塞线程。

3.计算处理时间

这部分主要用到了QueryPerformanceCounter 和 QueryPerformanceFrequency API。这里会涉及到API与C#交互的问题。代码如下:

```
[csharp]
01. [DllImport("kernel32.dll")]
02. private static extern bool QueryPerformanceCounter(ref long lpPerformanceCount);
03. [DllImport("kernel32.dll")]
04. private static extern bool QueryPerformanceFrequency(ref long lpFrequency);
```

4.对图像进行缩放处理

这部分是也是写在UserControlPictureBox类中。由于用户拖入的图片尺寸可能很大, 显示在PicttrueBox中虽说可以进行缩放显示, 但得到的Image对象还是原来图片你的尺寸, 所以为了提高转换效率, 就要对图片进行等比例缩放。核心代码如下:

```
[cpp]
01. Bitmap bitmap = new Bitmap(newWidth, newHeight, oldImage.PixelFormat);
02. Graphics g = Graphics.FromImage(bitmap);
03. g.Clear(Color.Transparent);
04. g.DrawImage(oldImage, new RectangleF(0, 0, newWidth, newHeight));
05. return Image.FromHbitmap(bitmap.GetHbitmap());
```

其中newWidth, newHeight是新的图片的尺寸, 这两个值的得到很简单。

5.总体说明

界面上有很多按钮, 其实每个按钮的事件处理程序都是一个, 我在程序中定义了一个枚举:

```
[csharp]
01. public enum ImageEffect
02. {
03.     GrayScale = 0, // 黑白
04.     Film = 1, // 底片
05.     Relief = 2, // 浮雕
06.     Soften = 3, // 柔化
```

快速回复



```

07.     Sharpen     = 4,      // 锐化
08.     Canvas      = 5,      // 油画
09. }

```

在按钮处理程序中根据不同的按钮ID，给ImageEffectManager类的ChangeEffect方法传递不同的参数。  
ChangeEffect方法的定义如下：

```

[csharp]
01. public void ChangeEffect(ImageEffect effect)
02. {
03.     switch(effect)
04.     {
05.         case ImageEffect .GrayScale :
06.             break;
07.     }
08. }

```

6. 代码

ImageEffectManager.cs类 图像效果管理

```

[csharp]
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Text;
05. using System.Drawing;
06. namespace ImageEffectSample.Classes
07. {
08.     public enum ImageEffect
09.     {
10.         GrayScale = 0,      // 黑白
11.         Film       = 1,      // 底片
12.         Relief     = 2,      // 浮雕
13.         Soften     = 3,      // 柔化
14.         Sharpen    = 4,      // 锐化
15.         Canvas     = 5,      // 油画
16.     }
17.     public class ImageEffectManager
18.     {
19.         private Bitmap newBitmap = null;
20.         private Bitmap oldBitmap = null;
21.         #region Properties
22.
23.         public Bitmap ConvertedBitmap
24.         {
25.             get { return this.newBitmap; }
26.         }
27.         public Bitmap OriginalBitmap
28.         {
29.             set
30.             {
31.                 DisposeBitmap();
32.                 this.oldBitmap = value;
33.             }
34.         }
35.         public Size PixelSize
36.         {
37.             get
38.             {
39.                 if (this.oldBitmap != null)
40.                 {
41.                     GraphicsUnit unit = GraphicsUnit.Pixel;
42.                     RectangleF bounds = this.oldBitmap.GetBounds(ref unit);
43.                     return new Size(
44.                         (int)bounds.Width, (int)bounds.Height);
45.                 }
46.                 return new Size(0, 0);
47.             }
48.         }
49.         #endregion

```

```
50.         #region Methods
51.
52.         public ImageEffectManager()
53.         {
54.         }
55.         public void DisposeBitmap()
56.         {
57.             if (this.newBitmap != null)
58.             {
59.                 this.newBitmap.Dispose();
60.                 this.newBitmap = null;
61.             }
62.         }
63.         public void ChangeEffect(ImageEffect effect)
64.         {
65.             if (null == this.oldBitmap)
66.             {
67.                 return;
68.             }
69.             Size size = PixelSize;
70.             int width = size.Width;
71.             int height = size.Height;
72.             this.DisposeBitmap();
73.             this.newBitmap = new Bitmap(width, height);
74.             switch (effect)
75.             {
76.                 case ImageEffect.GrayScale:
77.                     MakeGrayScale(width, height);
78.                     break;
79.                 case ImageEffect.Film:
80.                     MakeFilmEffect(width, height);
81.                     break;
82.                 case ImageEffect.Relief:
83.                     MakeReliefEffect(width, height);
84.                     break;
85.                 case ImageEffect.Soften:
86.                     MakeSoftenEffect(width, height);
87.                     break;
88.                 case ImageEffect.Sharpen:
89.                     MakeSharpenEffect(width, height);
90.                     break;
91.             }
92.         }
93.         private void MakeGrayScale(int width, int height)
94.         {
95.             Color c;
96.             for (int x = 0; x < width; x++)
97.             {
98.                 for (int y = 0; y < height; y++)
99.                 {
100.                     c = this.oldBitmap.GetPixel(x, y);
101.                     ///////////////////////////////////////////////////
102.                     //
103.                     // Average
104.                     // int value = (c.R + c.G + c.B) / 3;
105.                     //
106.                     ///////////////////////////////////////////////////
107.                     // Weighted average
108.                     int value = (int)(0.7 * c.R) +
109.                         (int)(0.2 * c.G) + (int)(0.1 * c.B);
110.                     this.newBitmap.SetPixel(x, y,
111.                         Color.FromArgb(c.A, value, value, value));
112.                 }
113.             }
114.         }
115.         private void MakeFilmEffect(int width, int height)
116.         {
117.             Color c;
118.             for (int x = 0; x < width; x++)
119.             {
120.                 for (int y = 0; y < height; y++)
121.                 {
122.                     c = this.oldBitmap.GetPixel(x, y);
123.                     this.newBitmap.SetPixel(x, y,
124.                         Color.FromArgb(c.A, 255 - c.R,
125.                             255 - c.G, 255 - c.B));
126.                 }
127.             }
128.         }
```

[^ 返回顶部](#)[快速回复](#)

[^ 返回顶部](#)

```
129.         private void MakeReliefEffect(int width, int height)
130.         {
131.             Color c1;
132.             Color c2;
133.             int r, g, b = 0;
134.             for (int x = 0; x < width - 1; x++)
135.             {
136.                 for (int y = 0; y < height - 1; y++)
137.                 {
138.                     c1 = this.oldBitmap.GetPixel(x, y);
139.                     c2 = this.oldBitmap.GetPixel(x + 1, y + 1);
140.                     r = Math.Abs(c1.R - c2.R + 128);
141.                     g = Math.Abs(c1.G - c2.G + 128);
142.                     b = Math.Abs(c1.B - c2.B + 128);
143.                     r = (r > 255) ? 255 : ((r < 0) ? 0 : r);
144.                     g = (g > 255) ? 255 : ((g < 0) ? 0 : g);
145.                     b = (b > 255) ? 255 : ((b < 0) ? 0 : b);
146.                     this.newBitmap.SetPixel(x, y,
147.                         Color.FromArgb(c1.A, r, g, b));
148.                 }
149.             }
150.         }
151.         private void MakeSoftenEffect(int width, int height)
152.         {
153.             // The template of Gauss
154.             int[] Gauss = { 1, 2, 1, 2, 4, 2, 1, 2, 1 };
155.             Color pixel;
156.             for (int x = 1; x < width - 1; x++)
157.             {
158.                 for (int y = 1; y < height - 1; y++)
159.                 {
160.                     int r = 0, g = 0, b = 0;
161.                     int Index = 0;
162.                     for (int col = -1; col <= 1; col++)
163.                     {
164.                         for (int row = -1; row <= 1; row++)
165.                         {
166.                             pixel =
167.                                 this.oldBitmap.GetPixel(x + row, y + col);
168.                             r += pixel.R * Gauss[Index];
169.                             g += pixel.G * Gauss[Index];
170.                             b += pixel.B * Gauss[Index];
171.                             Index++;
172.                         }
173.                     }
174.                     r /= 16;
175.                     g /= 16;
176.                     b /= 16;
177.                     //处理颜色值溢出
178.                     r = r > 255 ? 255 : r;
179.                     r = r < 0 ? 0 : r;
180.                     g = g > 255 ? 255 : g;
181.                     g = g < 0 ? 0 : g;
182.                     b = b > 255 ? 255 : b;
183.                     b = b < 0 ? 0 : b;
184.                     this.newBitmap.SetPixel(x - 1, y - 1,
185.                         Color.FromArgb(r, g, b));
186.                 }
187.             }
188.         }
189.         private void MakeSharpenEffect(int width, int height)
190.         {
191.             Color pixel;
192.             //拉普拉斯模板
193.             int[] Laplacian = { -1, -1, -1, -1, 9, -1, -1, -1, -1 };
194.             for (int x = 1; x < width - 1; x++)
195.             {
196.                 for (int y = 1; y < height - 1; y++)
197.                 {
198.                     int r = 0, g = 0, b = 0;
199.                     int Index = 0;
200.                     for (int col = -1; col <= 1; col++)
201.                     {
202.                         for (int row = -1; row <= 1; row++)
203.                         {
204.                             pixel = this.oldBitmap.GetPixel(x + row, y + col);
205.                             r += pixel.R * Laplacian[Index];
206.                             g += pixel.G * Laplacian[Index];
207.                             b += pixel.B * Laplacian[Index];
```

[快速回复](#)

```
208.         Index++;
209.     }
210. }
211. //处理颜色值溢出
212. r = r > 255 ? 255 : r;
213. r = r < 0 ? 0 : r;
214. g = g > 255 ? 255 : g;
215. g = g < 0 ? 0 : g;
216. b = b > 255 ? 255 : b;
217. b = b < 0 ? 0 : b;
218. this.newBitmap.SetPixel(x - 1, y - 1,
219.     Color.FromArgb(r, g, b));
220. }
221. }
222. }
223. private void MakeCanvasEffect(int width, int height)
224. {
225. }
226. #endregion
227. }
228. }
```

[^ 返回顶部](#)

TimeCounter.cs类，用于计算处理时间

```
[csharp]
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Text;
05. using System.Runtime.InteropServices;
06. namespace ImageEffectSample.Classes
07. {
08.     public class TimeCounter
09.     {
10.         private static long startCount = 0;
11.         private static long elapsedCount = 0;
12.         #region Properties
13.         public static float Seconds
14.         {
15.             get
16.             {
17.                 long freq = 0;
18.                 float retValue = 0.0f;
19.                 QueryPerformanceFrequency(ref freq);
20.                 if (freq != 0)
21.                 {
22.                     retValue = (float)elapsedCount / (float)freq;
23.                 }
24.                 return retValue;
25.             }
26.         }
27.         #endregion
28.         #region Methods
29.         public static void Start()
30.         {
31.             startCount = 0;
32.             QueryPerformanceCounter(ref startCount);
33.         }
34.         public static void Stop()
35.         {
36.             long stopCount = 0;
37.             QueryPerformanceCounter(ref stopCount);
38.             elapsedCount = (stopCount - startCount);
39.         }
40.         #endregion
41.         #region Import API
42.         [DllImport("kernel32.dll")]
43.         private static extern bool QueryPerformanceCounter(
44.             ref long lpPerformanceCount);
45.         [DllImport("kernel32.dll")]
46.         private static extern bool QueryPerformanceFrequency(
47.             ref long lpFrequency);
48.         #endregion
49.     }
50. }
```

[快速回复](#)

51. }

返回顶部

UserControlPictureBox.cs类 封装了图像缩放，拖拽，显示等功能

```
[csharp]
01. using System;
02. using System.Collections.Generic;
03. using System.ComponentModel;
04. using System.Drawing;
05. using System.Data;
06. using System.Linq;
07. using System.Text;
08. using System.Windows.Forms;
09. using System.Drawing.Imaging;
10. namespace ImageEffectSample.UserControls
11. {
12.     public partial class UserControlPictureBox : UserControl
13.     {
14.         private bool m_dragDropEnable = true;
15.         private bool m_isOriginalSize = false;
16.         #region Properties
17.         public PictureBox PictureBoxObject
18.         {
19.             get { return this.pictureBox; }
20.         }
21.         public bool DragDropEnable
22.         {
23.             get { return this.m_dragDropEnable; }
24.             set { m_dragDropEnable = value; }
25.         }
26.         public bool IsOriginalSize
27.         {
28.             get { return this.m_isOriginalSize; }
29.             set { m_isOriginalSize = value; }
30.         }
31.         #endregion
32.         #region Methods
33.
34.         public UserControlPictureBox()
35.         {
36.             InitializeComponent();
37.             this.pictureBox.AllowDrop = true;
38.             this.pictureBox.SizeMode = PictureBoxSizeMode.Zoom;
39.             this.pictureBox.DragDrop +=
40.                 new DragEventHandler(PictureBox_DragDrop);
41.             this.pictureBox.DragEnter +=
42.                 new DragEventHandler(PictureBox_DragEnter);
43.         }
44.         public void DisposeImage()
45.         {
46.             if (this.pictureBox.Image != null)
47.             {
48.                 this.pictureBox.Image.Dispose();
49.                 this.pictureBox.Image = null;
50.             }
51.         }
52.         private Image ZoomImage(Image oldImage)
53.         {
54.             if (null == oldImage)
55.             {
56.                 return null;
57.             }
58.             int width = oldImage.Width;
59.             int height = oldImage.Height;
60.             int newWidth = 0;
61.             int newHeight = 0;
62.             float ratioXY = (float)width / (float)height;
63.             if ((width > this.pictureBox.Width) ||
64.                 (height > this.pictureBox.Height))
65.             {
66.                 if (ratioXY >= 1.0)
67.                 {
68.                     newWidth = this.pictureBox.Width;
69.                     newHeight = (int)((float)newWidth / ratioXY) + 1;
70.                 }
71.                 else
```

快速回复

```
72.         {
73.             newHeight = this.pictureBox.Height;
74.             newWidth = (int)(newHeight * ratioXY);
75.         }
76.         Bitmap bitmap = new Bitmap(newWidth,
77.             newHeight, oldImage.PixelFormat);
78.         Graphics g = Graphics.FromImage(bitmap);
79.         g.Clear(Color.Transparent);
80.         g.DrawImage(oldImage,
81.             new RectangleF(0, 0, newWidth, newHeight));
82.         return Image.FromHbitmap(bitmap.GetHbitmap());
83.     }
84.     return oldImage;
85. }
86. #endregion
87. #region PictureBox drag and drop events
88. private void PictureBox_DragEnter(object sender, DragEventArgs e)
89. {
90.     bool bflag = e.Data.GetDataPresent(DataFormats.FileDrop);
91.     e.Effect = bflag && this.m_dragDropEnable ? e.Effect =
92.         DragDropEffects.Copy : DragDropEffects.None;
93. }
94. private void PictureBox_DragDrop(object sender, DragEventArgs e)
95. {
96.     if (e.Data.GetDataPresent(DataFormats.FileDrop))
97.     {
98.         String[] strfileNames =
99.             (String[])e.Data.GetData(DataFormats.FileDrop);
100.        Image dragImage = Image.FromFile(strfileNames[0]);
101.        if (dragImage != null)
102.        {
103.            this.DisposeImage();
104.            this.pictureBox.Image =
105.                m_isOriginalSize ? dragImage : ZoomImage(dragImage);
106.        }
107.    }
108. }
109. #endregion
110. }
111. }
```

### ImageEffectMainWindow.cs类，主窗体

```
[csharp]
01. using System;
02. using System.Collections.Generic;
03. using System.ComponentModel;
04. using System.Data;
05. using System.Drawing;
06. using System.Linq;
07. using System.Text;
08. using System.Windows.Forms;
09. using ImageEffectSample.Classes;
10. namespace ImageEffectSample.Windows
11. {
12.     public partial class ImageEffectMainWindow : Form
13.     {
14.         ImageEffectManager effectManager = null;
15.         public ImageEffectMainWindow()
16.         {
17.             InitializeComponent();
18.             this.effectManager = new ImageEffectManager();
19.             this.convertedPictureBox.DragDropEnable = false;
20.         }
21.         private void CheckBoxOriginalSize_CheckedChanged(object sender, EventArgs e)
22.         {
23.             this.originalPictureBox.IsOriginalSize =
24.                 this.checkBoxOriginalSize.Checked;
25.         }
26.         private void Button_Click(object sender, EventArgs e)
27.         {
28.             this.textBoxTime.Text = "";
29.             Button button = sender as Button;
30.             if (button != null)
31.             {
```



32.                   TimeCounter.Start();

33.                   Image img = this.originalPictureBox.PictureObject.Image;

34.                   if (img != null)

35.                   {

36.                         effectManager.OriginalBitmap = new Bitmap(img);

37.                         switch (button.Name)

38.                         {

39.                             case "btnGrayscaleEffect":

40.                                 effectManager.ChangeEffect(ImageEffect.GrayScale);

41.                                 break;

42.                             case "btnFileEffect":

43.                                 effectManager.ChangeEffect(ImageEffect.Film);

44.                                 break;

45.                             case "btnReliefEffect":

46.                                 effectManager.ChangeEffect(ImageEffect.Relief);

47.                                 break;

48.                             case "btnSoftenEffect":

49.                                 effectManager.ChangeEffect(ImageEffect.Soften);

50.                                 break;

51.                             case "btnSharpenEffect":

52.                                 effectManager.ChangeEffect(ImageEffect.Sharpen);

53.                                 break;

54.                         }

55.                         this.convertedPictureBox.DisposeImage();

56.                         this.convertedPictureBox.PictureObject.Image =

57.                             effectManager.ConvertedBitmap;

58.                         TimeCounter.Stop();

59.                         this.textBoxTime.Text =

60.                             String.Format("Elapsed time: {0} s",

61.                                 TimeCounter.Seconds.ToString());

62.                         }

63.                   }

64.           }

65.

66.   private void ClearImagesButton\_Click(object sender, EventArgs e)

67.   {

68.         this.convertedPictureBox.DisposeImage();

69.         this.originalPictureBox.DisposeImage();

70.         this.textBoxTime.Text = "";

71.   }

72. }

73. }

返回顶部



顶

0

踩

0

- 上一篇 x86程序读取64位系统注册表失败解决方案
- 下一篇 Windows文件检索之——接口设计

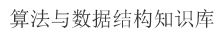
我的同类文章

VC++ (22)

快速回复

- [^ 返回顶部](#)

更多文章



9462 关注 | 2263 收录

- 使用C#开发信息管理系统
- 值得收藏的机器学习资源
- C#开发微信订阅号、服务号视频教程
- 计算机视觉机器学习图形学等学习资源网站
- .NET平台和C#编程从入门到精通
- 深度学习 资料整理
- 疯狂的 C#——快速入门
- 深度学习资源
- 《C语言/C++学习指南》加密解密篇（安全相关算法）
- GDAL学习总结



上海单身公寓广告

[查看评论](#)

1楼 [tianyawp123](#) 2016-09-08 10:31发表 







您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

## 核心技术类目

全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack		
VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	
BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS	Unity
Splashtop	UML	components	Windows	Mobile	Rails	QEMU	KDE	Cassandra	CloudStack			
FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo			
Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr		
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap							

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

 网站客服
  杂志客服
  微博客服
  webmaster@csdn.net
  400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2016, CSDN.NET, All Rights Reserved



 快速回复